

ALGORITHMIC AND HARDNESS RESULTS FOR SOME GRAPH OPTIMIZATION PROBLEMS

DOCTORAL THESIS

By

GOPIKA SHARMA

(2017MAZ0007)



**DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF TECHNOLOGY ROPAR
RUPNAGAR, PUNJAB 140001, INDIA**

March, 2024

ALGORITHMIC AND HARDNESS RESULTS FOR SOME GRAPH OPTIMIZATION PROBLEMS

A Thesis Submitted

In Partial Fulfilment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

by

**GOPIKA SHARMA
(2017MAZ0007)**

Under the Guidance of

DR. ARTI PANDEY



**Department of Mathematics
Indian Institute of Technology Ropar
March, 2024**

To my family (parents and bhai bhabhi), who never give up on me.

DECLARATION OF ORIGINALITY

I hereby declare that the work which is being presented in the thesis entitled **Algorithmic and Hardness Results for Some Graph Optimization Problems** has been solely authored by me. It presents the result of my own independent investigation/research conducted during the time period from January 5, 2018 to March 14, 2024 under the supervision of Dr. Arti Pandey, Assistant Professor, Department of Mathematics, Indian Institute of Technology Ropar. To the best of my knowledge, it is an original work, both in terms of research content and narrative, and has not been submitted or accepted elsewhere, in part or in full, for the award of any degree, diploma, fellowship, associateship, or similar title of any university or institution. Further, due credit has been attributed to the relevant state-of-the-art and collaborations (if any) with appropriate citations and acknowledgments, in line with established ethical norms and practices. I also declare that any idea/data/fact/source stated in my thesis has not been fabricated/ falsified/ misrepresented. All the principles of academic honesty and integrity have been followed. I fully understand that if the thesis is found to be unoriginal, fabricated, or plagiarized, the Institute reserves the right to withdraw the thesis from its archive and revoke the associated Degree conferred. Additionally, the Institute also reserves the right to appraise all concerned sections of society of the matter for their information and necessary action (if any). If accepted, I hereby consent for my thesis to be available online in the Institute's Open Access repository, inter-library loan, and the title & abstract to be made available to outside organizations.

Gopikasharma
Signature

Name: Gopika Sharma

Entry Number: 2017MAZ0007

Program: Ph.D.

Department: Mathematics

Indian Institute of Technology Ropar

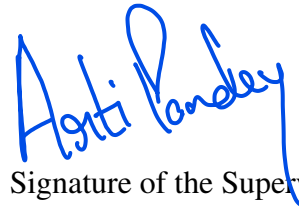
Rupnagar, Punjab 140001

Date: March 14, 2024

CERTIFICATE

This is to certify that the thesis entitled **Algorithmic and Hardness Results for Some Graph Optimization Problems**, submitted by **Gopika Sharma** for the award of the degree of **Doctor of Philosophy** of Indian Institute of Technology Ropar, is a record of bonafide research work carried out under my guidance and supervision. To the best of my knowledge and belief, the work presented in this thesis is original and has not been submitted, either in part or full, for the award of any other degree, diploma, fellowship, associateship or similar title of any university or institution.

In my opinion, the thesis has reached the standard fulfilling the requirements of the regulations relating to the Degree.



Signature of the Supervisor

Name: Dr. Arti Pandey

Department: Mathematics

Indian Institute of Technology Ropar

Rupnagar, Punjab 140001

Date: March 14, 2024

ACKNOWLEDGEMENT

The acquisition of this PhD has been a profoundly transformative experience for me, and I owe my success to the generous support and direction of many people. I am deeply grateful for their help and would like to express my heartfelt gratitude.

First and foremost I extend my sincere gratitude to my Ph.D. advisor Dr. Arti Pandey who deserves my most humble thanks before anyone else. I am thankful to her for her invaluable pieces of advice, continuous support, and patience during my Ph.D. study. She created environments where I may succeed despite my academic shortcomings. She provided me with the highest level of care, support, and treatment, and I owe her a lot for that.

I would like to thank my DC members Prof. Manoranjan Mishra, Dr. S. C. Martha, Dr. G. Sankara Raju Kosuru, and Dr. Sam Darshi for their continuous guidance and support. They all motivated me towards research throughout my Ph.D. I am grateful to all the faculty members of the Department of Mathematics, IIT Ropar for providing me with a warm environment and lifelong memories. I am thankful to our office assistant Mr. Neeraj Sharma for helping me with technical needs and Ms. Jaspreet Kaur for her support in all the official work needed.

I am grateful to the Ministry of Education, Government of India and IIT Ropar for financial support. I also thank IIT Ropar for providing exclusive travel grants to attend conferences in abroad and India which really helped me in bringing new ideas to my research and making connections for future research. I am thankful that IIT Ropar library has access to certain prestigious journals. I express my sincere thanks to all my collaborators: Prof. Boštjan Brešar, Prof. M. A. Henning, and, Mr. Michael C. Wigal for providing significant input in my research work. I also want to thank Nivedit Jain and Rashi Kamra for their valuable contributions to my research work. I thank my fellow research group members Vikash, Kusum, and Kaustav for fruitful discussions and dinner outings. I appreciate each one of them for being good friends and for sharing memorable times with me in addition to being my professional colleagues. I am grateful to my senior Dr. Vikash for all the discussions and guidance he provided me whenever I needed them the most. I also thank Kusum and Kaustav for all the productive discussions and continuous support.

I feel fortunate to have many friends at IIT Ropar. I thank all of them for making the stay at IIT Ropar a memorable experience for a lifetime. I want to thank Neha (cheeta), Neha (Paaro), Abhishek,

Shobhit, Ankit, Aman, Satinder, Shubhashini, Gargi, Kapil, Neelam, Ankita Mishra, and Karan for treating me not just a friend, but a family member and sharing wonderful memories. I enjoyed endless (sometimes senseless) talks, memorable trips with each one of them and all dinner outings that are unforgettable. This journey would have been much more difficult without this family cum group of friends. My friends Paaro and Cheeta deserve a wholehearted thank you for the many amazing moments we have shared as the “teen titliya” group. I also thank my first roommate at IIT Ropar Anamika Chhabra for treating me like her younger sister in the initial days of this journey. I thank in particular my friend Sakshi Pandey, who supported me through every ups and downs of my Ph.D. journey. She helped me in getting through a lot of challenging circumstances, personal or professional along the way.

Words cannot express my gratitude and love towards all of my friends from the department- Sonam, Monika, Aditi Jain, Niharika, Ankita Gupta, and Kusum. Without quality time spent with these lab mates, my stay at IIT Ropar would not have been memorable. All the celebrations held in our labs are unforgettable to me. I thank Abhijeet, Taranjot, Akriti, Himanshu, Priya, Ramandeep, Shubhendu, Surya, and, Sahil. I admire how kind they were to me and the pleasant memories they gave me. At last, I thank all my friends from the institute and fellow labmates of the Department of Mathematics whom I missed mentioning above.

I also thank my friends Ankita Dewan, Rohit, Navneet, Shubham, Akash, Amit, and Saroj from other departments for sharing many wonderful memories with me. My friends Priya, Shristi, Juhi, Khushboo, Nidhi, Saumya, Shachi di, Pragya, Aditi, Swati, and Himani never gave up on me and I am grateful to them for the same. My special thanks to Vivek Selvan in particular for always being there to hear all of my views as they emerged throughout the course of my Ph.D. I owe a lot to my friend Veekesh for motivating me whenever I felt disheartened.

Most importantly, I express my deepest appreciation to my family. Their efforts and sacrifices have been a driving force behind my success, and I am forever grateful to them. I thank my Meha bhabhi and brother Kanav Sharma for their endless love, support, and care. I am also grateful to my in-laws and other family members for their constant encouragement and support. I owe a special debt of gratitude to my mother-in-law for her blessings and good wishes. Last but not least, I wish to express my deepest thanks to my fiancé, Abhishek Bhanot for his constant encouragement, care, and love.

Finally, I would like to thank God, the Almighty, who has granted me countless blessings, knowledge, and opportunities so that I have been finally able to accomplish the thesis.

Lay Summary

Graphs can be used to model the real world situations, where we have a collection of objects that have pairwise relationships. For example, a computer network can be represented as a graph, where each server is a vertex and the connections between servers are edges. In this way, many real world optimization problems can be modeled as graph optimization problems. Graph optimization problems seek to optimize a quantity associated with the graph, such as maximizing or minimizing it. In this thesis, we have studied five important graph optimization problems, which are listed below.

1. MAXIMUM INTERNAL SPANNING TREE Problem
2. MINIMUM EDGE TOTAL DOMINATING SET Problem
3. GRUNDY (DOUBLE) DOMINATION Problem
4. MAXIMUM WEIGHTED EDGE BICLIQUE Problem
5. NEIGHBOR-LOCATING COLORING Problem

These problems have applications in various areas including social networks, computer networks, telephone switching networks, bi-clustering analysis, etc. Unfortunately, solving these problems for general graphs is challenging, and we cannot realistically expect to find optimal solutions in a practical amount of time. Therefore, we focus on understanding the complexity of these problems for important subclasses of graphs. The study of such graph problems constrained to particular graph classes has attracted considerable interest and is an active research area in algorithmic graph theory. Specifically, we try to see whether a problem under consideration can be solved in a reasonable amount of time in some special cases or not. We also try to find some combinatorial bounds on optimization parameters in terms of other important graph parameters. These bounds often help in designing approximation algorithms for a problem that outputs a near-optimal solution in a reasonable amount of time.

Abstract

Many real world optimization problems can be modeled as graph optimization problems. However, several graph optimization problems that are practically significant are NP-hard for general graphs, and hence, it is unlikely to find exact solutions in polynomial-time. One approach to tackle this is to study the problem for some restricted graph classes, as most of the time graphs obtained by modeling real world problems exhibit some special properties. Many of the researchers are working to design polynomial-time algorithms for NP-hard graph optimization problems for restricted graph classes. This thesis considers five important graph optimization problems, namely (i) MAXIMUM INTERNAL SPANNING TREE Problem, (ii) MINIMUM EDGE TOTAL DOMINATING SET Problem, (iii) GRUNDY (DOUBLE) DOMINATION Problem, (iv) MAXIMUM WEIGHTED EDGE BICLIQUE Problem and (v) NEIGHBOR-LOCATING COLORING Problem.

A spanning tree of a graph G containing the maximum number of internal vertices among all spanning trees of G is called a *maximum internal spanning tree* (MIST) of G . The MAXIMUM INTERNAL SPANNING TREE problem is to find a MIST for a given graph G . For some special graph classes, we provide linear-time algorithms to compute a MIST and relate the number of internal vertices in a MIST with an important graph parameter.

For a graph $G = (V, E)$ without an isolated edge, a set $D \subseteq E$ is called an *edge total dominating set* (ETD-set) of G if every edge $e \in E$ is adjacent to at least one edge of D . For a given graph G with no isolated edges, the MINIMUM EDGE TOTAL DOMINATING SET Problem is to find an ETD-set of G with minimum cardinality. We study the problem in subclasses of bipartite graphs and chordal graphs. We prove some results from the approximation aspects. We also discuss the complexity difference between the MINIMUM EDGE DOMINATING SET problem and the MINIMUM EDGE TOTAL DOMINATING SET problem.

A sequence $S = (v_1, v_2, \dots, v_k)$ is a *dominating sequence* if $N[v_i] \setminus \bigcup_{j=1}^{i-1} N[v_j] \neq \emptyset$ holds for every $i \in \{2, \dots, k\}$ and $\widehat{S} = \{v_1, v_2, \dots, v_k\}$ is a dominating set of G . The GRUNDY DOMINATION problem is to find a dominating sequence of maximum length for a given graph G . Moreover, a sequence $S = (v_1, v_2, \dots, v_k)$ is a *double dominating sequence* if (i) for each $i \in [k]$, the vertex v_i dominates at least one vertex of G that was dominated at most once by the previous vertices of S and (ii), $\widehat{S} = \{v_1, v_2, \dots, v_k\}$ is a double dominating set of G . The GRUNDY DOUBLE

DOMINATION problem is to find a double dominating sequence of maximum length for a given graph G . We find some graph classes for which the problems are solvable in polynomial-time. Additionally, we identify certain graph classes for which these problems are NP-hard.

Given a weighted graph $G = (V, E, w)$, where each edge $e \in E$ has a weight $w(e) \in \mathbb{R}$, the MAXIMUM WEIGHTED EDGE BICLIQUE problem is to find a biclique C of G such that the sum of the weights of edges of C is maximum. We obtain some algorithmic results when $w(e) \in \mathbb{R}^+$.

A proper coloring of a graph is called a *neighbor-locating coloring* if, for any two vertices of the same color, the sets of colors of their neighborhoods are different. Given a graph G , the NEIGHBOR-LOCATING COLORING problem requires assigning a color to each vertex of G such that the coloring is a neighbor-locating coloring and the number of colors used is minimized. We provide some algorithmic and combinatorial results for this parameter in some special graph classes.

Contents

DECLARATION OF ORIGINALITY	vii
CERTIFICATE	vii
ACKNOWLEDGEMENT	ix
LAY SUMMARY	xiii
ABSTRACT	xv
CONTENTS	xvii
LIST OF FIGURES	xxi
LIST OF TABLES	xxiii
1 Introduction	1
1.1 Basic Notations and Definitions	2
1.1.1 Graph Theoretic Notations	2
1.1.2 Algorithmic Preliminaries	4
1.2 Graph Classes Studied in the Thesis	9
1.2.1 Bipartite Graphs	9
1.2.2 Trees	12
1.2.3 Chordal Graphs	13
1.2.4 Cographs	16
1.2.5 Cactus Graphs	17
1.2.6 Co-bipartite Graphs	17
1.2.7 Co-chain Graphs	18
1.3 Summary of the Problems Studied in the Thesis	18
1.3.1 Maximum Internal Spanning Tree Problem	18
1.3.2 Minimum Edge Total Dominating Set Problem	19
1.3.3 Grundy (Double) Domination Problem	21
1.3.4 Maximum Weighted Edge Biclique Problem	22
1.3.5 Neighbor-Locating Coloring Problem	23
1.4 Structure and Contributions of the Thesis	25

2	Maximum Internal Spanning Tree	27
2.1	Introduction	27
2.2	Polynomial-Time Algorithms	30
2.2.1	Block and Cactus Graphs	30
2.2.2	Cographs	33
2.2.3	Bipartite Permutation Graphs	36
2.2.4	Chain Graphs	52
2.3	Relationship between $Opt(G)$ and $ E(P^*) $	57
2.3.1	Block/Cactus Graph	58
2.3.2	Bipartite Permutation Graph	59
2.3.3	Chain Graph and Cographs	60
2.4	Summary	60
3	Edge Total Dominating Set	63
3.1	Introduction	63
3.2	Efficient Algorithms	64
3.2.1	Chain Graphs	64
3.2.2	Split Graphs	72
3.2.3	Proper Interval Graphs	74
3.3	APX-completeness and Approximation Algorithm	80
3.4	Complexity Difference Between Edge Domination and Edge Total Domination . . .	85
3.5	Summary	89
4	Grundy (Double) Dominating Sequence	91
4.1	Introduction	91
4.2	Dominating Sequences	96
4.2.1	NP-completeness results	97
4.2.1.1	Bipartite Graphs	97
4.2.1.2	Co-bipartite Graphs	101
4.2.2	Algorithm for Chain Graphs	102
4.3	Double Dominating Sequences	117
4.3.1	NP-completeness Results	118
4.3.1.1	Split Graphs	118
4.3.1.2	Bipartite Graphs	120
4.3.1.3	Co-bipartite Graphs	124
4.3.2	Efficient Algorithms	127
4.3.2.1	Threshold Graphs	128
4.3.2.2	Chain Graphs	130
4.4	Summary	140

5	Maximum Weighted Edge Biclique	141
5.1	Introduction	141
5.2	NP-completeness Result	142
5.3	Polynomial Time Algorithms	143
5.3.1	Bipartite Permutation Graphs	144
5.3.2	Chain Graphs	149
5.4	Summary	153
6	Neighbor-Locating Coloring	155
6.1	Introduction	155
6.2	Bounds	156
6.2.1	Chain Graphs	158
6.2.2	Proper Interval Graphs	162
6.2.3	Co-bipartite Graphs	164
6.3	Approximation Algorithm	168
6.4	Neighbor-Locating Coloring vs Vertex Coloring	169
6.5	Summary	172
7	Conclusion and Future Directions	173
7.1	Contributions	173
7.2	Future Directions	175
	REFERENCES	179

List of Figures

1.1	A hierarchy of well-studied graph classes.	7
1.2	A hierarchy of bipartite graphs and its subclasses.	10
1.3	Graph $K_{3,2}$	10
1.4	A Chain Graph.	11
1.5	A permutation graph G on 5 vertices with permutation $(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5) = (5, 4, 2, 1, 3)$	12
1.6	A Bipartite Permutation Graph.	12
1.7	A Tree.	13
1.8	A hierarchy of chordal graphs and its subclasses.	13
1.9	A Split Graph	14
1.10	A Threshold Graph	14
1.11	An example of an interval graph and its corresponding interval representation.	15
1.12	A proper interval graph and its corresponding interval representation.	15
1.13	A block graph.	16
1.14	A Cograph.	17
1.15	A Cactus Graph.	17
1.16	A Co-bipartite Graph.	18
1.17	A Co-chain Graph.	18
2.1	Illustrating a cograph and its cotree	34
2.2	Optimal path cover of G containing more than 1 path components	35
2.3	P ends at X side, Q starts from X side	54
2.4	P ends at Y side, Q starts from Y side	54
2.5	P ends at Y side, Q starts from X side	55
2.6	P ends at X side, Q starts from Y side and ends at X side	56
2.7	P ends at X side, Q starts from Y side and ends at Y side	56
2.8	Examples showing that bounds given by Theorem 2.2 and Theorem 2.15 for chain graphs are tight	58
2.9	Graph G_{20} , its optimal path cover P^* and its MIST T	59

2.10	Graph G_{25} , its optimal path cover P^* from Algorithm 3 and its MIST T from Algorithm 2	60
3.1	A chain graph G	71
3.2	A split graph $G = (I, K, E)$	73
3.3	A PIG with no cut vertex and its interval representation.	79
3.4	A gadget H_{v_i} corresponding to a vertex v_i of V	81
3.5	A GP_8 graph obtained from C_4	86
3.6	A GP_3 graph obtained from C_4	88
4.1	Some Graph Classes.	96
4.2	Construction of bipartite graph G from the hypergraph H	98
4.3	Construction of a co-bipartite graph G' from a graph G	101
4.4	Construction of bipartite graph G from the hypergraph H	119
4.5	Construction of bipartite graph G from the hypergraph H	121
4.6	Construction of co-bipartite graph G' from the graph G	125
5.1	An illustration to the construction of H from G	143
5.2	An example of Bipartite Permutation Graph.	146
5.3	An example of Chain Graph.	150

List of Tables

3.1	An illustration of the Algorithm 5.	71
3.2	An illustration of the Algorithm 6.	74
3.3	An illustration of the Algorithm 7.	79

Chapter 1

Introduction

Graph theory plays a significant role in modeling pairwise relationships between discrete objects. In a variety of domains, including computer science, communication networks, electrical networks, and social sciences, graph theory has strong applications. The famous “Königsberg bridge problem” is considered to be the origin of graph theory. This problem takes us back to the 18th century. At that time, seven bridges crossed the river Pregel in the town of Königsberg. People of the town were trying to solve a puzzle: Is it possible to walk through the town using every bridge just once and return to the starting point at the end? By converting the problem into a graph problem, Leonhard Euler was able to find a solution in 1736, however, he gave the slightly disappointing conclusion that no such walk existed. His solution was based only on the actual configuration of bridges but essentially he proved the first theorem in graph theory. In the years that followed, graph theory saw significant development and is now considered to be one of the major areas of mathematics.

Numerous problems in the actual world can be modeled as graph-theoretic problems. For instance, consider the following situation. We have a transport network in which a node represents a crossroad and a link between two crossroads represents a road. We want to install surveillance cameras on some of the crossroads so that every road is monitored by a camera. The goal is to locate the minimum number of cameras in order to minimize the cost. This situation can be modeled as a graph $G = (V, E)$, where the vertices of the graph are crossroads and edges are roads. To achieve the goal, we must select a set $S \subseteq V$ of the minimum cardinality such that every edge has at least one end vertex in the set S . In this way, a real world problem is modeled as a graph-theoretic problem.

An optimization problem is a problem in which the goal is to choose the best solution from a range of possibilities while adhering to specific constraints. The best (optimal) solution is determined by maximizing or minimizing a quantity within the scope of these constraints. The constraints in these problems act as restrictions that must be considered. Over time, various types of optimization problems have emerged, and specialized techniques have developed to tackle them.

Several real-life optimization problems can be seen as graph-theoretic problems. A graph optimization problem is an optimization problem where our objective is to optimize (maximize or minimize) a quantity associated with a graph. For instance, in the transport network example discussed

above, our goal was to minimize the number of vertices in a subset of the vertex set of the graph with respect to some conditions. Graph optimization problems include an extensive number of well-known problems in graph theory and operations research. They appear to be theoretically very interesting problems in addition to being ones with practical relevance. The development of efficient techniques and numerical algorithms to solve such problems has been an active area of research for many decades. In fact, of Karp's 21 problems in the paper ([50]), 10 problems are some versions of graph optimization problems, while most of the other 11 problems can be naturally formulated on graphs. In this thesis, we have studied five graph optimization problems, namely:

1. MAXIMUM INTERNAL SPANNING TREE Problem
2. MINIMUM EDGE TOTAL DOMINATING SET Problem
3. GRUNDY (DOUBLE) DOMINATION Problem
4. MAXIMUM WEIGHTED EDGE BICLIQUE Problem
5. NEIGHBOR-LOCATING COLORING Problem

Before presenting a brief summary of the above listed problems, we first discuss some basic notations and definitions that are used throughout the thesis.

1.1 Basic Notations and Definitions

In this section, we go through some crucial graph theoretic and algorithmic notations that are employed throughout the thesis.

1.1.1 Graph Theoretic Notations

For graph theoretic notations, we follow [10]. The set of integers $\{1, 2, \dots, k\}$ is denoted by $[k]$. Throughout the thesis, n denotes the number of vertices in the graph and m denotes the number of edges in the graph. A graph is said to be *non-trivial* if it contains at least two vertices. Let $G = (V, E)$ be a graph. An edge between the vertices u and v is denoted by uv . For an edge $e = uv \in E$, u and v are called the *end vertices* of e . A vertex u is said to be *incident* with an edge e if u is one of the end vertices of e . Two vertices u and v of G are called *adjacent* to each other or *neighbors* of each other if $uv \in E$. Two edges $e, f \in E$ are also called *adjacent*, if they share an end vertex. For a vertex

$v \in V$, the set of all neighbors of v is denoted by $N_G(v)$ and this set is called the *open neighborhood* of v in G . The set $N_G[v] = N_G(v) \cup \{v\}$ is called the *closed neighbourhood* of v in G . When the graph is clear from the context, we use $N(v)$ and $N[v]$ to denote $N_G(v)$ and $N_G[v]$, respectively. For a non-empty set $A \subseteq V$, the *open neighborhood of A* is denoted by $N_G(A)$ and is given by $N_G(A) = \bigcup_{v \in A} N_G(v)$ whereas the set $N_G[A] = N_G(A) \cup A$ is known as the *closed neighborhood of A* . Two vertices $u, v \in V$ are called *open (closed) twins* if $N(u) = N(v)$ ($N[u] = N[v]$). The *degree* of a vertex $u \in V$ is defined as the number of vertices adjacent to u and is denoted by $d_G(u)$. We use the symbols $\delta(G) = \min\{d_G(v) | v \in V\}$ and $\Delta(G) = \max\{d_G(v) | v \in V\}$ to denote the *minimum* and *maximum* degree of G , respectively. A vertex v is called a *pendant* vertex or a *leaf* if $d_G(v) = 1$. For a pendant vertex v , the vertex adjacent to v is called the *support* vertex of v and is denoted by $s(v)$. A vertex v is called an *internal* vertex of G if $d_G(v) \geq 2$. A vertex v is called an *isolated* vertex of G if $d_G(v) = 0$. An edge is called an *isolated* edge if no other edge is adjacent to it.

For a set $A \subseteq V$, $G \setminus A$ represents the graph obtained by removing the vertices of the set A and all edges having at least one end vertex in A , from G . If $A = \{u\}$, we write $G \setminus u$, instead of $G \setminus \{u\}$. For a set $S \subseteq V$, $G[S]$ denotes the subgraph of G induced by S , whose vertex set is S and two vertices $x, y \in S$ are adjacent in $G[S]$ if and only if $xy \in E$. Similarly, for a set $D \subseteq E$, $G[D]$ denotes the subgraph of G induced by the set of edges present in D , with vertex set $\{v \in V(G) : v \text{ is incident with at least one edge } e \in D\}$ and edge set D . A subgraph of G is called a *spanning subgraph* if the subgraph contains all the vertices of G .

A *path* on k vertices in G , denoted by P_k , is a sequence of vertices (x_1, x_2, \dots, x_k) such that $x_i x_{i+1} \in E$ for each $i \in \{1, 2, \dots, k-1\}$. The length of a path P_k is defined as $|V(P_k)| - 1 = k - 1$. A path containing all vertices of the graph is called a *Hamiltonian* path in G . A graph is said to be *connected* if there exists a path between every pair of vertices in the graph. A graph is *disconnected* if it is not connected. For a disconnected graph, every maximal connected subgraph is called a *connected component* or simply a *component* of the graph. In a connected graph, a vertex v is called a *cut vertex* of the graph if the removal of the vertex v makes the graph disconnected. A connected graph having no cut vertex is called a *biconnected* graph.

A set $X \subseteq V$ is called an *independent set* if no two vertices of X are adjacent in G . An independent set in G with the maximum number of vertices is called a *maximum independent set* in G and the cardinality of such a set is called the *independence number* of G , denoted by $\alpha(G)$. Two edges are called *independent* if they are not adjacent to each other. A set $V_c \subseteq V$ is called a *vertex cover*

of G if every edge $e \in E$ has at least one end vertex in V_c . A vertex cover of G with the minimum number of vertices is called a *minimum vertex cover* of G and the cardinality of such a set is called the *vertex cover number* of G , denoted by $\beta(G)$. A set $K \subseteq V$ is called a *clique* in G if for any two vertices $x, y \in K$, $xy \in E$. In this case, we also say that K induces a clique in G . A clique in G with the maximum number of vertices is called a *maximum clique* in G and the cardinality of such a set is called the *clique number* of G , denoted by $\omega(G)$. A set $D \subseteq V$ is called a *dominating set* of G if for every vertex $u \in V \setminus D$, $N(u) \cap D \neq \emptyset$. By “ u dominates v ” or “ v is dominated by u ”, we mean that $v \in N[u]$. A dominating set of G with the minimum number of vertices is called a *minimum dominating set* of G and the cardinality of such a set is called the *domination number* of G , denoted by $\gamma(G)$. A set $D \subseteq V$ is called a *double dominating set* of G if for every vertex $u \in V$, $|N[u] \cap D| \geq 2$. A subset M of the edge set E is called a *matching* if no two edges in M have a common end vertex. A vertex $v \in V$ is said to be *saturated* by a matching M if v is incident with an edge in M .

A spanning subgraph of G is called a *path cover* if each component of the subgraph is a path. A path (x_1, x_2, \dots, x_n) together with the additional edge x_1x_n is called a *cycle* on n vertices and is denoted by C_n . A cycle containing all vertices of the graph is called a *Hamiltonian cycle* in G . We call the graph G a *Hamiltonian graph* if there exists a Hamiltonian cycle in G . Let $x, y \in V(G)$. The *distance* between x and y in the graph G , denoted by $d_G(x, y)$, is the length of the shortest path between x and y . A graph G is called a *complete graph* if $E(G) = \{uv : u, v \in V\}$ and is denoted by K_n .

In this thesis, we consider only simple, undirected, and connected graphs.

1.1.2 Algorithmic Preliminaries

Here, we go over a few concepts related to algorithms and definitions used in this thesis. For these terminologies, we follow [9] and [27]. When the input to an algorithm is a graph, the *input size* is described by the numbers of vertices and edges in the graph. If G is a graph with n vertices and m edges, then the input size is $n + m$. The number of operations or “steps” executed by an algorithm is known as its *running time* and is denoted by $O(\text{input size})$. An *efficient algorithm* is an algorithm whose running time is bounded by a polynomial in its input size. We denote the running time of an efficient algorithm by $O(\text{poly}(\text{input size}))$, where $\text{poly}((\text{input size}))$ denotes a polynomial function in the input size. A *polynomial-time algorithm* is another term used to refer to an efficient algorithm.

The study of *computational complexity* aims to classify various computational problems according to how effectively they can be solved. In general, two types of computational problems are considered: optimization problems, which require an optimal solution (maximum or minimum value), and decision problems, which only require a “YES” or “NO” answer. Formally, an *optimization problem* is defined as follows:

Definition 1.1 ([9]). An optimization problem Π is a quadruple $(I_\Pi, \text{SOL}_\Pi, m_\Pi, \text{goal}_\Pi)$, where:

1. I_Π is the set of instances of the problem Π ,
2. SOL_Π is a function that transforms each input instance of Π to its set of feasible solutions,
3. m_Π is defined for pairs (x, y) such that $x \in I_\Pi$ and $y \in \text{SOL}_\Pi(x)$ giving the measure of the function. For every pair (x, y) , $m_\Pi(x, y)$ equals a positive integer which is the value of the feasible solution y ,
4. $\text{goal}_\Pi \in \{\text{MAX}, \text{MIN}\}$ describes whether Π is a maximization or a minimization problem.

For example, consider the MAXIMUM INDEPENDENT SET (MIS) problem. Given a graph G , the MIS problem asks to find an independent set of G of the maximum cardinality. The MIS problem is an optimization problem. Each element of the quadruple for this problem is defined as follows:

1. $I = \{G = (V, E) \mid G \text{ is a graph}\}$,
2. $\text{SOL}(G) = \{I \mid I \text{ is an independent set of } G\}$,
3. $m(G, V_c) = |I|$,
4. $\text{goal} = \text{MAX}$.

On the other hand, a *decision problem* Π have a set of instances I_Π and for a given instance $I \in I_\Pi$ there is a query associated with I whose answer is either YES (True) or NO (False). For example, the HAMILTONIAN CYCLE problem is a decision problem. For this problem, the set of instances is the set of all graphs G . The query associated with every instance G of the problem is “Is G a Hamiltonian graph?”

These decision and optimization problems are categorized in complexity theory according to computational complexity. (i) Class P: It contains all the decision problems for which there exists a

polynomial-time algorithm to solve it. (ii) Class NP: It contains all the decision problems which can be verified in polynomial-time. By verification, we mean that given an instance $I \in I_\Pi$ of a problem Π and a certificate $C(I)$ having a polynomial size in terms of the size of I , there exists a verification algorithm that takes I and $C(I)$ as input and in polynomial-time returns “YES” if and only if I is a YES instance. The definitions of the P and NP classes indicate that P is included within NP. The famous millennium hypothesis “ $P \neq NP$ ” is significant to note here because it is still unsolved as of this writing.

In complexity theory, the idea of *reductions* is presented to group the problems into categories. Suppose that we have a procedure that transforms any instance I_A of a decision problem A into some instance I_B of another decision problem B satisfying the following characteristics:

- The time taken by the transformation is polynomial.
- The answers of both the instances are same. That is, I_A is a “yes” instance if and only if I_B is a “yes” instance.

We call such a procedure a polynomial-time reduction from the problem A to the problem B . In other words, we say that the problem A is polynomially reducible to the problem B and we write $A \leq_P B$ to denote this.

A decision problem Π is said to be *NP-hard* if every problem in class NP is polynomially reducible to Π . A problem in NP is called an *NP-complete* problem if it is also NP-hard. According to traditional beliefs, unless $P = NP$, it is not possible to devise a polynomial-time algorithm for an NP-complete problem.

Unfortunately, most of the graph problems are NP-hard for general cases. But due to the importance of these problems in real world situations, various approaches are taken to deal with the intractability of an NP-hard problem. It can be done in a variety of ways. The most natural way is to identify a set of instances for which the problem can be solved efficiently. In terms of graphs, this is equivalent to determine the graph classes for which the problem can be solved in polynomial-time. Some of the well-studied graph classes are given in Figure 1.1. Another approach is to design and analyze efficient approximation algorithms. Here, we choose to settle for a good approximate solution to the problem rather than computing the optimal solution. The reliability of an approximation algorithm can be measured by its approximation ratio.

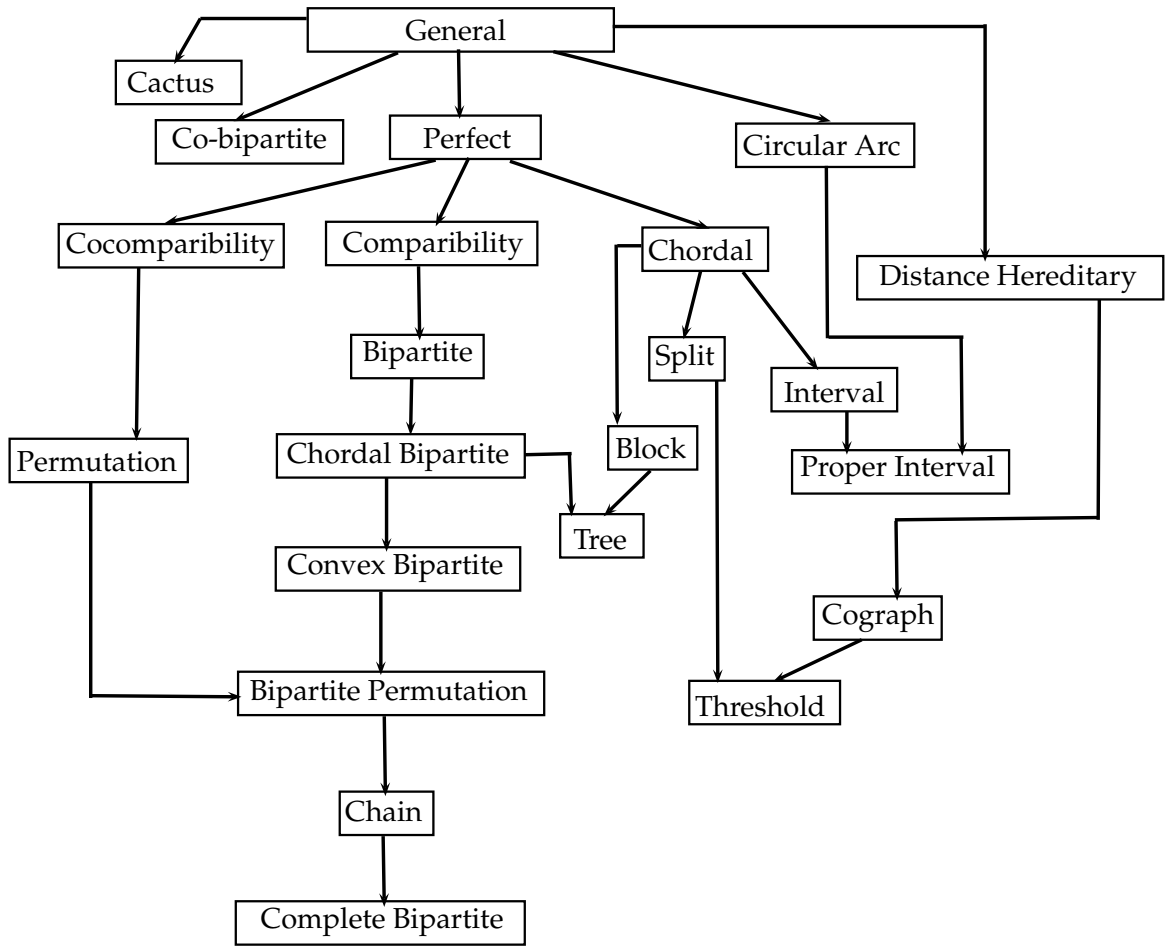


FIGURE 1.1: A hierarchy of well-studied graph classes.

Definition 1.2 ([90]). **r -Approximation Algorithm:** Let Π be an optimization problem. A polynomial-time algorithm, that returns a solution for an instance $I \in I_\Pi$ such that the value of that solution lies within a factor of r of the value of an optimal solution of I , is called an r -approximation algorithm for the problem Π .

Here, the real number r is called the *approximation ratio* or *performance ratio* of the associated approximation algorithm. Let Π be an optimization (minimization/maximization) problem and A_Π be an r -approximation algorithm for the problem Π . We denote the value of the solution for an instance $I \in I_\Pi$, returned by A_Π by $A_\Pi(I)$ and the value of an optimal solution by $opt(I)$. If Π is a minimization problem then

$$r = \max \left\{ \frac{A_\Pi(I)}{opt(I)} \mid I \in I_\Pi \right\}$$

Otherwise, if Π is a maximization problem then

$$r = \max \left\{ \frac{\text{opt}(I)}{A_{\Pi}(I)} \mid I \in I_{\Pi} \right\}.$$

An optimization problem Π is said to be r -approximable if there exists an r -approximation algorithm for Π .

One can divide the optimization problems into different classes based on the performance ratio. An optimization problem Π belongs to the class APX, if there exists a polynomial-time r -approximation algorithm, where r is a constant. Depending on the performance ratio, there are various other classes. We refer to [90] for the details of these complexity classes of optimization problems. In this thesis, other classes are not of interest to us, thus we will omit them from our discussion.

Given a complexity class C of optimization problems and a type of reduction, an optimization problem Π is said to be C -hard with respect to this type of reduction if each problem in C has a reduction of the given type to Π . In addition, if Π belongs to the complexity class C , then Π is said to be C -complete. In this thesis, we are focused on APX-hard problems and the type of reduction we used is called L-reduction. Formally, an L-reduction is defined as follows:

Definition 1.3. *L-reduction*:

Let $P_1 = (I_{P_1}, \text{SOL}_{P_1}, m_{P_1}, \text{MIN})$ and $P_2 = (I_{P_2}, \text{SOL}_{P_2}, m_{P_2}, \text{MIN})$ be two minimization problems. Let $f: I_{P_1} \rightarrow I_{P_2}$ be a polynomial-time function that transforms each instance of P_1 to an instance of P_2 . We say that the function f is an L-reduction if there exist $a > 0$ and $b > 0$ such that for any instance $I \in I_{P_1}$, the following holds:

1. $\min f(I) \leq a \cdot \min(I)$.
2. For every feasible solution $y \in \text{SOL}_{P_2}(f(I))$, in polynomial time, we can find a solution $x \in \text{SOL}_{P_1}(I)$ such that $|\min(I) - m_{P_1}(I, x)| \leq b \cdot |\min(f(I)) - m_{P_2}(f(I), y)|$.

To show that an optimization problem $\Pi \in \text{APX}$ is APX-complete, we need to show the existence of an L-reduction from some known APX-hard problem to the problem Π .

Studying parameterized algorithms, heuristics, and meta-heuristics are some more strategies for handling NP-hard problems. In this thesis, these techniques have not been applied in particular. For

details on parameterized complexity, we refer [33]. Therefore, we omit the details of these approaches for handling the intractability of NP-hard problems.

1.2 Graph Classes Studied in the Thesis

In this section, we formally define all the graph classes discussed in later chapters of this thesis.

Note: Special property(ies) of these graphs will be explained in detail wherever needed.

1.2.1 Bipartite Graphs

A graph $G = (V, E)$ is called a *bipartite graph* if the vertex set V can be partitioned into two independent sets X and Y . The pair (X, Y) is called the *bipartition* of G . In a bipartite graph G , we see that for any $e \in E$, one of the end vertex of e belongs to the set X and other end vertex belongs to the set Y . Typically a bipartite graph is denoted by $G = (X, Y, E)$. There are several characterizations of bipartite graphs. The most commonly used characterization is: “A graph G is a bipartite graph if and only if G contains no odd cycle”.

Most of the optimization problems which are NP-hard for general graphs remain NP-hard for bipartite graphs as well. This forces researchers to explore the complexity status of an optimization problem in subclasses of bipartite graphs having some special structure. The well known subclasses of bipartite graphs in the literature are: *perfect elimination bipartite graphs*, *chordal bipartite graphs*, *convex bipartite graphs*, *bipartite permutation graphs* and *chain graphs*. A hierarchy of bipartite graphs and its subclasses is shown in Figure 1.2. In this thesis, we have worked on bipartite graphs and some of its subclasses which are defined below.

1. **Complete Bipartite Graphs:** A bipartite graph $G = (X, Y, E)$ is called a *complete bipartite graph* if for any $x \in X$ and $y \in Y$, $xy \in E$. If $|X| = n_1$ and $|Y| = n_2$, then the complete bipartite graph G is denoted by K_{n_1, n_2} . Most of the optimization problems are trivially solvable for complete bipartite graphs. An example of a complete bipartite graph is shown in Figure 1.3.
2. **Chain Graphs:** A bipartite graph $G = (X, Y, E)$ is called a *chain graph* if the neighborhoods of the vertices of G form a chain; that is, the vertices of X and Y can be linearly ordered, say $X = \{x_1, x_2, \dots, x_{n_1}\}$ and $Y = \{y_1, y_2, \dots, y_{n_2}\}$ such that $N(x_1) \subseteq N(x_2) \subseteq \dots \subseteq N(x_{n_1})$ and $N(y_1) \supseteq N(y_2) \supseteq \dots \supseteq N(y_{n_2})$. The ordering $\alpha_0 =$

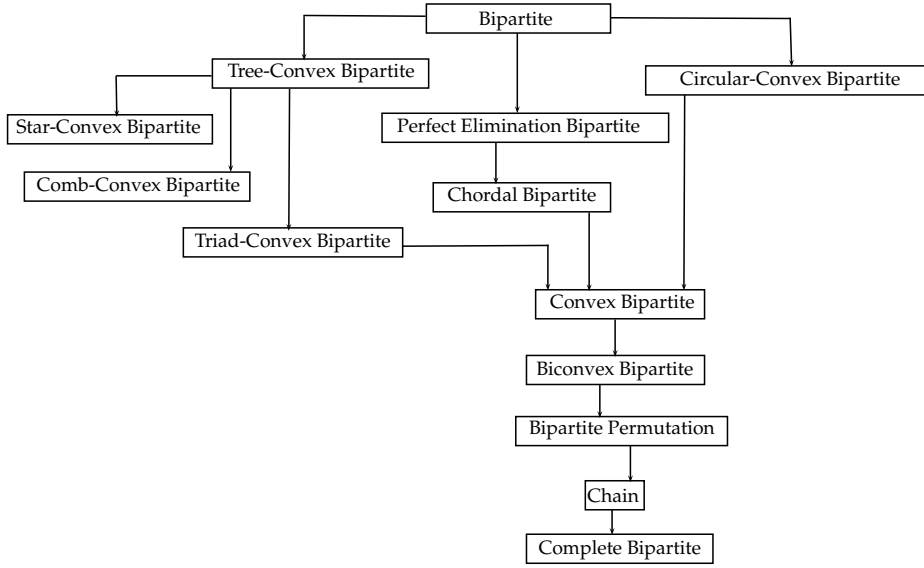
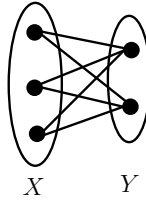


FIGURE 1.2: A hierarchy of bipartite graphs and its subclasses.

FIGURE 1.3: Graph $K_{3,2}$.

$(x_1, x_2, \dots, x_{n_1}, y_1, y_2, \dots, y_{n_2})$ is called a chain ordering of G . Given a chain graph G , a chain ordering of G can be computed in linear-time [45]. Throughout this thesis, chain ordering of a chain graph is denoted by $(x_1, x_2, \dots, x_{n_1}, y_1, y_2, \dots, y_{n_2})$. Sometimes, we may also write (O_X, O_Y) to denote this, where $O_X = (x_1, x_2, \dots, x_{n_1})$ and $O_Y = (y_1, y_2, \dots, y_{n_2})$.

An equivalence relation \sim on the vertex set of a chain graph $G = (X, Y, E)$ can be defined as follows. Two vertices of G are related with the relation \sim if and only if they are open twins. In this way, we get a partition of vertex subsets X and Y of G . We name the parts obtained for the X side by X_1, X_2, \dots, X_{k_1} and the parts obtained for the Y side are denoted by Y_1, Y_2, \dots, Y_{k_2} . Here, we assign the labels to the parts so that they satisfy $N(X_1) \subset N(X_2) \subset \dots \subset N(X_{k_1})$ and $N(Y_1) \supset N(Y_2) \supset \dots \supset N(Y_{k_2})$. Denote the set $\{X_1, X_2, \dots, X_{k_1}\}$ by P_X and the set $\{Y_1, Y_2, \dots, Y_{k_2}\}$ by P_Y . We call the sets P_X and P_Y obtained by the relation \sim the *twin partition* of X and Y respectively.

Note that such a partition can be obtained for any bipartite graph. For chain graphs, this partition can be viewed as a partition satisfying specific useful properties.

Proposition 1. In a chain graph G , let \sim be the relation defined on $V(G)$ as discussed above, then $k_1 = k_2$.

Proof. We have that $N(X_1) \subset N(X_2) \subset \dots \subset N(X_{k_1})$ and $N(Y_1) \supset N(Y_2) \supset \dots \supset N(Y_{k_2})$ holds true. For any $i < j$, $N(X_i)$ is a proper subset of $N(X_j)$. So, let $y \in N(X_j)$ such that $y \notin N(X_i)$. Since the graph is connected, this gives rise to at least two sets in P_Y . Hence, we get that $k_2 \geq k_1$. Similarly, $N(Y_i) \supset N(Y_j)$ gives $k_1 \geq k_2$ implying that $|P_X| = k_1 = k_2 = |P_Y|$. \square

Suppose $k_1 = k_2 = k$. Then the following result can be observed easily.

Proposition 2. For a chain graph $G = (X, Y, E)$, let $P_X = \{X_1, X_2, \dots, X_k\}$ and $P_Y = \{Y_1, Y_2, \dots, Y_k\}$ be the twin partition of X and Y respectively. Then $N(X_i) = \cup_{j=1}^i Y_j$ and $N(Y_i) = \cup_{j=i}^k X_j$.

A chain graph together with the partition obtained by the relation \sim is shown in Figure 1.4.

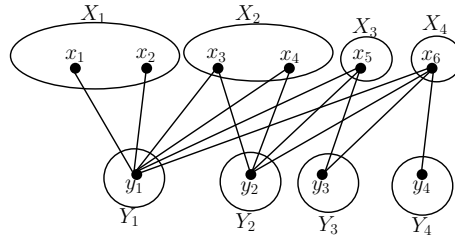


FIGURE 1.4: A Chain Graph.

3. **Bipartite Permutation Graphs:** A graph $G = (V, E)$ with $V = \{v_1, v_2, \dots, v_n\}$ is said to be a *permutation graph* if there is a permutation σ over $\{1, 2, \dots, n\}$ such that $v_i v_j \in E$ if and only if $(i - j)(\sigma^{-1}(i) - \sigma^{-1}(j)) < 0$. In other words, each vertex v in a permutation graph G is associated with a line segment l_v joining two points on two parallel lines L_1 and L_2 , which is called a line representation. Then, two vertices v and u are adjacent in G if and only if the corresponding line segments l_v and l_u are crossing. A permutation graph together with its line representation is shown in Figure 1.5.

A graph is a *bipartite permutation graph* if it is both a bipartite and permutation graph.

Several characterizations of bipartite permutation graphs are available in the literature. A *strong ordering* ($<_X, <_Y$) of a bipartite graph $G = (X, Y, E)$ consists of an ordering $<_X$ of X and an ordering $<_Y$ of Y , such that for all edges $ab, a'b'$, with $a, a' \in X$ and $b, b' \in Y$; if $a <_X a'$ and $b' <_Y b$, then ab' and $a'b$ are edges in G . A strong ordering of a bipartite permutation

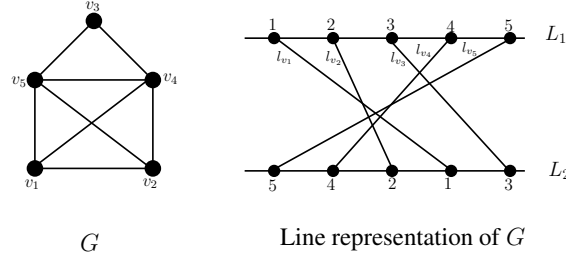


FIGURE 1.5: A permutation graph G on 5 vertices with permutation $(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5) = (5, 4, 2, 1, 3)$

graph can be computed in linear-time [86]. An ordering $<_X$ of X has the *adjacency property* if, for every vertex in Y , its neighbors in X are consecutive in $<_X$. The ordering $<_X$ has the *enclosure property* if, for every pair of vertices y, y' of Y with $N(y) \subseteq N(y')$, the vertices of $N(y') \setminus N(y)$ appear consecutively in $<_X$.

Authors in [46], proved that *a bipartite graph is a bipartite permutation graph if and only if it admits a strong ordering*. Furthermore, if we assume that the graph is connected, then we can say more.

Lemma 1.4. [46] *Let $(<_X, <_Y)$ be a strong ordering of a connected bipartite permutation graph $G = (X, Y, E)$. Then both $<_X$ and $<_Y$ have the adjacency property and the enclosure property.*

A bipartite permutation graph along with its strong ordering is shown in Figure 1.6. Throughout this thesis, we denote the strong ordering of vertices of G by $(<_X, <_Y)$.

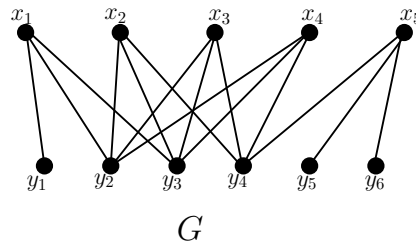


FIGURE 1.6: A Bipartite Permutation Graph.

1.2.2 Trees

A graph is called a *tree* if it is connected and contains no cycle. For a tree T , we have $|E(T)| = |V(T)| - 1$. Recall that a graph is bipartite if and only if it contains no odd cycle. As a tree contains no cycle at all, it is also a bipartite graph. An example of a tree is shown in Figure 1.7.

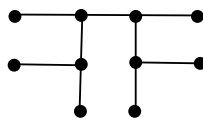


FIGURE 1.7: A Tree.

1.2.3 Chordal Graphs

A *chordal graph* is a graph that does not contain an induced cycle of length greater than 4. The class of chordal graphs is an important subclass of perfect graphs. The complexity of many NP-hard problems for general graphs is studied for the class of chordal graphs. Chordal graphs are well known due to the existence of a vertex ordering called “perfect elimination ordering”.

Let $G = (V, E)$ be a graph containing n vertices. A vertex $v \in V$ is called a *simplicial vertex* of G if the closed neighborhood of v induces a clique in G . There is another characterization of chordal graphs based on simplicial vertices: “a graph G is a chordal graph if and only if every induced subgraph of G has a simplicial vertex”. Further, an ordering $\alpha = (v_1, v_2, \dots, v_n)$ of vertices in V is called a *perfect elimination ordering* if the vertex v_i is a simplicial vertex in the induced graph $G[\{v_i, v_{i+1}, \dots, v_n\}]$. Fulkerson and Gross [38] characterized chordal graphs, and showed that a graph G is chordal if and only if it admits a perfect elimination ordering (PEO). We have studied the complexity of some graph parameters in the following subclasses of chordal graphs which are more structured. A hierarchy of chordal graphs and some of its subclasses are shown in Figure 1.8

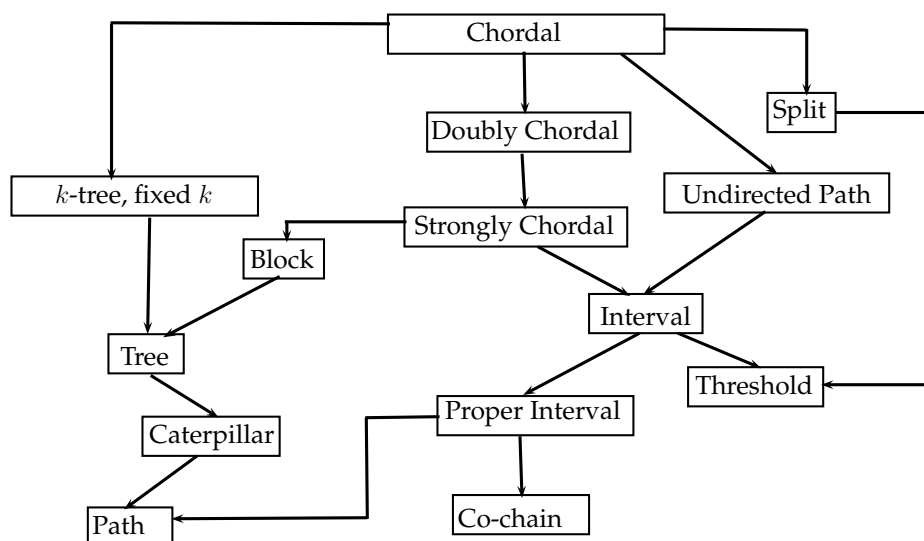


FIGURE 1.8: A hierarchy of chordal graphs and its subclasses.

1. **Split Graphs:** A graph $G = (V, E)$ is called a *split graph* if the vertex set V can be partitioned into two sets I and K such that I is an independent set of G and K is a clique in G . This is an important subclass of chordal graphs for which the complexity of the many optimization problems is studied in the literature. A split graph is shown in Figure 1.9.

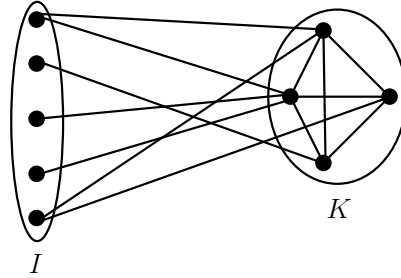


FIGURE 1.9: A Split Graph

2. **Threshold Graphs:** A *threshold graph* is a graph that can be constructed from the graph K_1 by repeated applications of the following two operations:
 - (a) Addition of a single isolated vertex to the graph.
 - (b) Addition of a single vertex to the graph such that it is adjacent to all the vertices added prior to that vertex.

A threshold graph is shown in Figure 1.10. In this figure, black vertices are added to the graph by the second operation and white vertices are added to the graph by the first operation. The labels of the vertices represent the order in which they are added to the graph. Threshold graphs

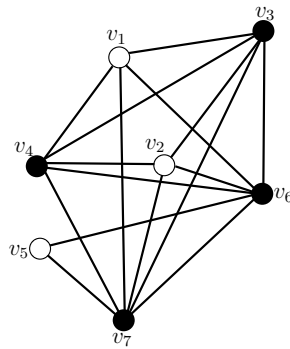


FIGURE 1.10: A Threshold Graph

were introduced first in the book [66]. Threshold graphs can be characterized in various ways. They can be viewed as special cases of some wider classes of graphs like cographs, split graphs, interval graphs, etc.

3. **Interval Graphs:** A graph $G = (V, E)$ is an *interval graph* if and only if there is a one-to-one correspondence between its vertex set V and a family of closed intervals $\mathcal{I} = \{I_k = [a_k, b_k]\}$ on the real line, such that two vertices of G are adjacent if and only if their corresponding intervals intersect. Such a family of intervals $\mathcal{I} = \{I_k = [a_k, b_k]\}$ is called an *interval representation* corresponding to the interval graph G . Booth and Lueker [15] designed a linear-time recognition algorithm for interval graphs. An example of an interval graph and its corresponding interval representation is shown in Figure 1.11.

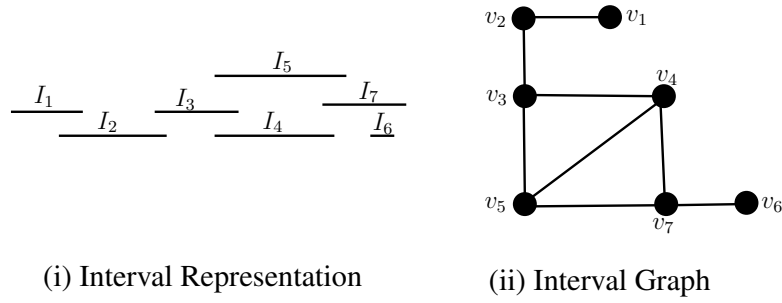


FIGURE 1.11: An example of an interval graph and its corresponding interval representation.

4. **Proper Interval Graph:** A *proper interval graph* is an interval graph such that no interval properly contains another in its interval representation. Due to its structural properties, various optimization problems are efficiently solvable for this graph class. A *unit interval graph* is an interval graph in which all the intervals in its interval representation are of unit length. It is known that class of unit interval graphs and proper interval graphs coincide [43]. An example of a proper interval graph and its corresponding interval representation is shown in Figure 1.12.

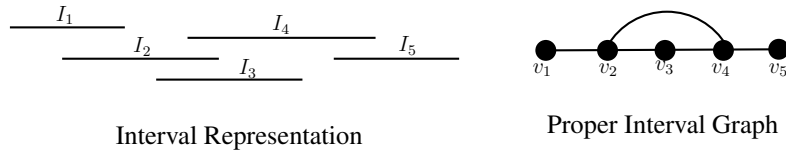


FIGURE 1.12: A proper interval graph and its corresponding interval representation.

For a vertex $v \in V(G)$, if $N(v)$ induces a complete subgraph of G , then v is called a *simplicial vertex* of G . A *perfect elimination ordering* (PEO) of G is an ordering $\alpha_0 = (v_1, v_2, \dots, v_n)$ of $V(G)$ such that v_i is a simplicial vertex of $G[\{v_i, v_{i+1}, \dots, v_n\}]$, for every $i \in [n]$. A PEO, $\alpha_0 = (v_1, v_2, \dots, v_n)$, of G is a *bicompatible elimination ordering* (BCO) if $\alpha_0^{-1} = (v_n, v_{n-1}, \dots, v_1)$, that is, the reverse of α_0 , is also a PEO of G . It is known that a graph G has

a BCO if and only if it is a proper interval graph [48]. Panda and Das proposed a linear-time algorithm to compute a BCO of a proper interval graph [76].

5. **Block Graphs:** For a graph G , a maximal induced subgraph of G without a cut vertex is called a *block* of G . Two blocks of a graph have at most one common vertex, that is, if $V(B_1) \cap V(B_2) \neq \emptyset$ for two arbitrary blocks of G then $|V(B_1) \cap V(B_2)| = 1$. A vertex $v \in V(B_1) \cap V(B_2)$ if and only if v is a cut vertex of G .

A graph G is called a *block graph* if each of its blocks is a clique. Note that an edge is also a clique so a tree is also a block graph. A complete graph G is also a block graph having exactly one block, which is G itself. An example of a block graph is shown in Figure 1.13.

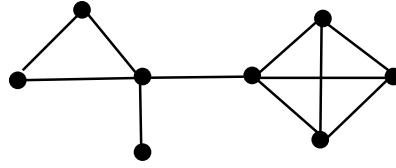


FIGURE 1.13: A block graph.

A block containing only one cut vertex is called an *end block*. Every block graph G has at least two end blocks if G is not isomorphic to a complete graph.

1.2.4 Cographs

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs such that $V_1 \cap V_2 = \emptyset$ and $E_1 \cap E_2 = \emptyset$. The *disjoint union* $G = (V, E)$, of G_1 and G_2 is a graph such that $V = V_1 \cup V_2$ and $E = E_1 \cup E_2$. The *complement* of a graph $G = (V, E)$ is a graph $\bar{G} = (V', E')$ where $V' = V$ and $E' = \{e : e \notin E\}$. The class of *cographs* is defined recursively as follows:

1. K_1 is a cograph.
2. The complement of a cograph is also a cograph.
3. The disjoint union of two cographs is also a cograph.

An example of a cograph is shown in Figure 1.14.

Cographs were independently identified under various names and were shown to have many characterizations. Cographs are exactly the P_4 restricted graphs. A cograph has a unique tree

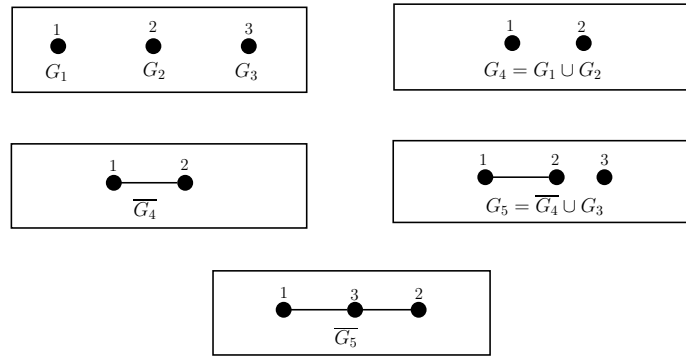


FIGURE 1.14: A Cograph.

representation associated with it. This tree, called a cotree, helps in designing fast polynomial time algorithms for various graph optimization problems in cographs.

1.2.5 Cactus Graphs

A graph is *cactus graph* if each of its block is either a cycle or an edge. These are graphs in which each edge is present in at most one cycle. Several researchers from different scientific fields are interested in cactus graphs because they have many applications in biology and computer science. An example of a cactus graph is shown in Figure 1.15.

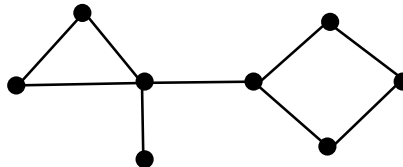


FIGURE 1.15: A Cactus Graph.

1.2.6 Co-bipartite Graphs

A graph $G = (V, E)$ is called a *co-bipartite graph* if its complement is a bipartite graph. In other words, these are graphs whose vertex set can be partitioned into at most two cliques. Most of graph optimization problems are NP-hard when the input is a co-bipartite graph. A co-bipartite graph is shown in Figure 1.16.

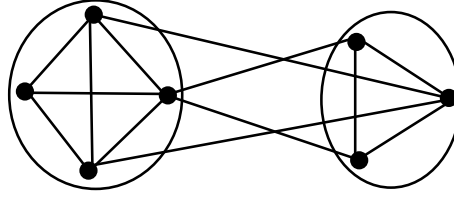


FIGURE 1.16: A Co-bipartite Graph.

1.2.7 Co-chain Graphs

A *co-chain graph* is a co-bipartite graph in which the neighborhoods of the vertices in each side can be linearly ordered with respect to inclusion. Complement of a chain graph is also a co-chain graph. An example of a co-chain graph is shown in Figure 1.17.

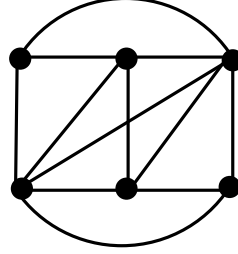


FIGURE 1.17: A Co-chain Graph.

1.3 Summary of the Problems Studied in the Thesis

In this thesis, we study five important graph optimization problems. We present all the necessary definitions, problem statements, the motivation behind the problem, and the literature review for each of the graph optimization problems in a sequential manner below.

1.3.1 Maximum Internal Spanning Tree Problem

A *spanning tree* of a graph G is a spanning subgraph of G which is also a tree. A spanning tree of G containing the maximum number of internal vertices among all the spanning trees of G is called a *maximum internal spanning tree* (MIST) of G . Given a graph G , the MAXIMUM INTERNAL SPANNING TREE (MIST) problem is to find a MIST of G .

The formal definition of the problem and its decision version are stated below:

MAXIMUM INTERNAL SPANNING TREE (MIST) Problem*Input:* A graph $G = (V, E)$.*Output:* A maximum internal spanning tree of G .**DECIDE MIST Problem***Input:* A graph $G = (V, E)$ and $k \in \mathbb{Z}^+$.*Question:* Does there exists a spanning tree T of G containing at least k internal vertices?

The MIST problem has theoretical significance because it generalizes the HAMILTONIAN PATH problem, a well-known NP-complete problem. Consequently, the Decide MIST problem is also NP-hard for general graphs. Since the HAMILTONIAN PATH problem is NP-hard, even for chordal graphs and chordal bipartite graphs, the Decide MIST problem also remains NP-hard for these graph classes. The MIST problem has been studied in the literature from an algorithmic point of view for many years. Several constant factor approximation algorithms have been devised for this problem, see [52, 58, 60, 80, 81]. In 2018, Chen et al. presented a $\frac{17}{13}$ -approximation algorithm which is the best approximation factor till now [25]. For cubic graphs, Salaman et al. designed a $\frac{6}{5}$ -approximation algorithm in 2008 [25]. A detailed survey on the history of approximation algorithms for the MIST problem can be found in [82]. Recently, Li et al. proved that the MIST problem is Max-SNP-hard [61].

To the best of our knowledge, the class of interval graphs is the first non-trivial graph class for which the Decide MIST problem can be solved in polynomial-time [57]. We look for some more graph classes for which the problem belongs to the class P and obtain some positive results.

1.3.2 Minimum Edge Total Dominating Set Problem

In a graph $G = (V, E)$, a subset $F \subseteq E$ is called an *edge dominating set* (ED-set) of G if, for each $e \in E$, either $e \in F$ or e is adjacent to an edge in F . The MINIMUM EDGE DOMINATING SET (Min-EDS) problem requires us to find an edge dominating set of minimum cardinality for a given graph G . The minimum cardinality of an edge dominating set of G is called the *edge domination number* of G , denoted by $\gamma'(G)$. In 1991, Kulli and Patwari [53] introduced the notion of *edge total domination* in graphs. For a graph, $G = (V, E)$ without an isolated edge, a set $D \subseteq E$ is called an *edge total dominating set*, abbreviated ETD-set, of G if every edge $e \in E$ is adjacent to at least one edge of D . An ETD-set with minimum cardinality is called a *minimum edge total dominating set* (min-ETD-set) of G . The cardinality of a min-ETD-set of G is called the *edge total domination*

number of G , denoted by $\gamma'_t(G)$. For a given graph G with no isolated edges, the MINIMUM EDGE TOTAL DOMINATING SET (Min-ETDS) problem is to find a min-ETD-set of G .

The Min-EDS problem can be defined as the task of keeping track of the state of communications occurring among nodes in a wireless ad hoc network. To understand this, assume there is a wireless ad hoc network. Any pair of nodes in such networks has the ability to start a packet exchange, and routing can take any path the network chooses. Here, the goal is to place monitoring devices at some nodes so that each communication is monitored. This task requires us to recognize a set of communication links (edges) such that every other edge is adjacent to at least one edge belonging to this set. By placing monitoring devices at the endpoints of each selected edge we can achieve our goal. If we call the corresponding graph G , then the set of selected edges is an edge dominating set of G . Due to cost considerations, we look for such sets with minimum size, that is, we seek minimum edge dominating sets. Moreover, there can be a failure of devices so we want a backup for each communication link. This situation can be handled by computing a min-ETD-set of G .

The formal definition of the problem and its decision version are stated below:

MINIMUM EDGE TOTAL DOMINATING SET (Min-ETDS) Problem

Input: A graph $G = (V, E)$ with no isolated edge.

Output: A min-ETD-set D of G .

DECIDE MIN-ETDS Problem

Input: A graph $G = (V, E)$ with no isolated vertices and $k \in \mathbb{Z}^+$.

Question: Does there exists an ETD-set D of G such that $|D| \leq k$?

The Min-ETDS problem and some of its variations gained considerable attention. Many researchers studied both algorithmic and combinatorial aspects of these problems, see [1, 16, 53, 70, 77, 78, 85, 89]. Specifically, Kulli and Patwari in 1991, computed $\gamma'_t(G)$ when G is restricted to paths, cycles, complete graphs, and complete bipartite graphs [53]. In 2014, Zhao et al. [92] proved that the decision version of the Min-ETDS problem is NP-complete for planar graphs with maximum degree 3. In the same article, they proved that it remains NP-complete for undirected path graphs, a subclass of chordal graphs. Recently, in 2020, Zhuo et al. [75] proved that the problem is NP-complete for bipartite graphs with maximum degree 3. On the positive side, the problem is linear-time solvable for trees. We investigate the complexity status of this problem in some other graph classes and we increase the size of the set of the graph classes for which the Decide Min-ETDS problem is in class P .

1.3.3 Grundy (Double) Domination Problem

A sequence $S = (v_1, v_2, \dots, v_k)$ of vertices of G is called a *dominating sequence* of G if (i) for each $i \in [k]$, the vertex v_i dominates at least one vertex of G that was not dominated by the previous vertices of S and, (ii) the set $\hat{S} = \{v_1, v_2, \dots, v_k\}$ is a dominating set of G . A dominating sequence of maximum length is called a *Grundy dominating sequence* of G . The GRUNDY DOMINATION (GD) problem is to find a Grundy dominating sequence of a given graph G . Further, a sequence $S = (v_1, v_2, \dots, v_k)$ of vertices of G is called a *double dominating sequence* of G if (i) for each $i \in [k]$, the vertex v_i dominates at least one vertex of G that was dominated at most once by the previous vertices of S and, (ii) the set $\hat{S} = \{v_1, v_2, \dots, v_k\}$ is a double dominating set of G . A double dominating sequence of maximum length is called a *Grundy double dominating sequence* of G . The GRUNDY DOUBLE DOMINATION (GD2) problem is to find a Grundy double dominating sequence of a given graph G .

One of the oldest and most well-researched areas in graph theory is domination which is known for many real world applications. Grundy dominating sequence of the graph somehow describes the worst scenario that can happen when a dominating set is built. When we adopt a greedy procedure to construct a dominating set of a graph, we pick vertices in the solution set one by one so that each chosen vertex enlarges the set of vertices dominated by the selected vertices. This procedure also motivates the notion of dominating sequences. A dominating set D in a graph can be seen as a set of vertices governing or monitoring the vertices in $V \setminus D$. Then the removal or failure of a vertex in D or of an arbitrary edge may become the reason for the set D to not be a dominating set anymore. If this is an undesirable situation, then it may be useful to increase the level of domination of each vertex, so that, even if a vertex or edge fails, the set D will still be a dominating set in G . This idea led researchers to introduce the concept of multiple domination. Therefore, the notion of double dominating sequences is also introduced [44].

The formal definition of the problems and its decision versions are stated below:

GRUNDY DOMINATION (GD) Problem

Input: A graph $G = (V, E)$.

Output: A Grundy dominating sequence S of G .

DECIDE GRUNDY DOMINATION (GDD) Problem

Input: A graph $G = (V, E)$ and $k \in \mathbb{Z}^+$.

Question: Does there exists a dominating sequence S of G such that $|\hat{S}| \geq k$?

GRUNDY DOUBLE DOMINATION (GD2) Problem

Input: A graph $G = (V, E)$ with no isolated vertices.

Output: A Grundy dominating sequence S of G .

DECIDE GRUNDY DOUBLE DOMINATION (GD2D) Problem

Input: A graph $G = (V, E)$ with no isolated vertices and $k \in \mathbb{Z}^+$.

Question: Does there exists a double dominating sequence S of G such that $|\hat{S}| \geq k$?

In 2014, Brešar et al. proved that the GDD problem is NP-complete for chordal graphs. They also proved that a Grundy dominating sequence of trees, cographs and split graphs can be computed in polynomial time (see [20]). Several combinatorial results have also been established for the parameter in the literature [20, 44]. Grundy domination number of a graph is also studied in some standard graph products [17]. Brešar et al. continued the study of this parameter and obtained further that the GDD problem can be solved in polynomial time for interval and sierpiński graphs [19]. Several authors explored this variant of dominating sequences in the literature (see [18, 72]). The Grundy double domination number of a tree T is exactly the number of vertices of T [44]. We investigate the complexity status of both problems in various graph classes and obtain some positive results and some hardness results.

1.3.4 Maximum Weighted Edge Biclique Problem

For a graph G , a complete bipartite subgraph of G is called a *biclique* of G . For a weighted graph $G = (V, E, w)$, where each edge $e \in E$ has a weight $w(e) \in \mathbb{R}$, a biclique H of G such that $\sum_{e \in E(H)} w(e)$ is maximum among all bicliques of G , is called a *maximum weighted edge biclique* of G . The **MAXIMUM WEIGHTED EDGE BICLIQUE (MWEB)** problem is to find a maximum weighted edge biclique of a given weighted graph $G = (V, E, w)$.

Finding bicliques in bipartite graphs has a long history of applications. One important application is found in the biclustering analysis of gene expression data. Suppose we have a set of genes and a set of conditions. Having this, we form a bipartite graph $G = (A, B, E)$. In this graph, A is the set of conditions, and B is the set of genes. For $u \in A, v \in B$, we have $(u, v) \in E$ iff v responds in condition u , that is, if the expression level of v changes significantly in u . The weight of an edge

somehow measures the level of significance corresponding to its endpoints. Here, a biclique of G represents a subset of genes that are co-regulated under a subset of conditions. Then a maximum weighted edge biclique of G corresponds to the most significant bicluster in the input data.

The formal problem statement and its decision version are stated as follows:

MAXIMUM WEIGHTED EDGE BICLIQUE (MWEB) Problem

Input: A weighted graph $G = (V, E, w)$, $w \in \mathbb{R}$.

Output: A maximum weighted edge biclique C of G .

DECIDE MAXIMUM WEIGHTED EDGE BICLIQUE (WEBD) Problem

Input: A weighted graph $G = (V, E, w)$, $w \in \mathbb{R}^+$ and a positive integer $k \in \mathbb{R}^+$.

Output: Does there exists a weighted edge biclique C in G having weight at least k ?

The MWEB problem was introduced in 1979 [40] and is well studied in the literature, see [30, 29, 47, 88]. The WEBD problem is NP-complete for general graphs and even for bipartite graphs [79]. Note that the MWEB problem is the generalized version of the MEB problem. So, all the hardness results for the MEB problem are also valid for the MWEB problem [36, 42]. Therefore, the MWEB problem is hard to approximate in bipartite graphs within n^δ for some $\delta > 0$ under certain assumptions such as random 4-SAT or 3-SAT hardness hypothesis. There exists a restricted version of the MWEB problem, namely the S -MWEB problem, where S is a subset of real numbers from which edge weights are taken and the input graph is a bipartite graph. In 2008, Tan ([88]) proved that for a wide range of choices of S , no polynomial time algorithm can approximate the S -MWEB problem within a factor of n^ϵ for some $\epsilon > 0$ unless $RP = NP$. We study the problem for some particular choices of S .

1.3.5 Neighbor-Locating Coloring Problem

A vertex coloring of a graph $G = (V, E)$ is called *proper coloring* if no two adjacent vertices receive the same color. The minimum number of colors required for a proper coloring of G is called the *chromatic number* of G . A proper coloring of G is said to be *neighbor-locating coloring* (NL-coloring) if for each pair of vertices $u, v \in V$ having the same color, the sets of colors assigned to the neighbors of u and v are different. The minimum number of colors required for an NL-coloring of G is called the *neighbor-locating chromatic number* of G and is denoted by $\chi_{NL}(G)$. A neighbor-locating coloring of G which uses $\chi_{NL}(G)$ colors is called a *minimum neighbor-locating coloring* of G . The NEIGHBOR-LOCATING COLORING (NLC) problem asks to find an NL-coloring of G which uses $\chi_{NL}(G)$ number of colors.

Suppose that a company manufactures n chemicals. There exist some pairs of these chemicals which are incompatible with each other. When two incompatible chemicals are in contact, there is a risk of explosion. The company wants to create partitions in its warehouse so that incompatible chemicals can be kept in separate compartments as a precaution. What is the least number of compartments into which the warehouse should be partitioned? This can be answered via vertex coloring. We form a graph G by taking a vertex for each chemical and we make two vertices adjacent iff the corresponding chemicals are incompatible with each other. Then the chromatic number of G will be the least number of compartments into which the warehouse should be partitioned. Moreover, if there is a desirable situation where the number of compartments should be in such a way that two chemicals kept in the same compartment can be distinguished on the basis of the incompatibility of chemicals, the neighbor-locating chromatic number will be useful.

The formal problem statement and its decision version are stated as follows:

NEIGHBOR-LOCATING COLORING (NLC) Problem

Instance: A graph $G = (V, E)$.

Solution: A minimum neighbor-locating coloring of G .

DECIDE NEIGHBOR-LOCATING COLORING (NLCD) problem

Input: A graph $G = (V, E)$ and a positive integer $k \in \mathbb{R}^+$.

Question: Does there exist a neighbor-locating coloring of G using at most k colors?

There is no hardness result on the NLC problem in the literature. Alcon et al. gave some bounds for the neighbor-locating chromatic number of a general graph [3]. In the same article, they examined the neighbor-locating chromatic number for some graph operations: the join and the disjoint union. Moreover, the neighbor-locating chromatic number of split graphs and mycielski graphs have been computed. Alcon et al. also established some bounds on the neighbor-locating chromatic number for unicyclic graphs and trees in another paper [4]. Alcon et al. gave the neighbor-locating chromatic number of paths, cycles, fans, and wheels [2]. In 2020, Alcon et al. characterized all graphs as having neighbor-locating chromatic numbers equal to n or $n - 1$, where n is the number of vertices in the graph [3]. In 2022, Mojdesht [69] studied the conjectures posed by Alcon et al. in [3]. We prove some more bounds for the neighbor-locating chromatic number in some restricted graph classes.

1.4 Structure and Contributions of the Thesis

The structure of the thesis is summarized as follows: the first chapter of the thesis is dedicated to introduction and literature survey. This chapter also provides all the relevant definitions and important theorems which are used in subsequent chapters of the thesis. The rest of the thesis is organized as follows.

Chapter 2: Maximum Internal Spanning Tree

In this chapter, we investigate the complexity of the MIST problem in various graph classes and obtain some positive results. We propose linear-time algorithms to compute a maximum internal spanning tree of cographs, block graphs, cactus graphs, chain graphs, and bipartite permutation graphs. The OPTIMAL PATH COVER problem, which asks to find a path cover of a given graph with the maximum number of edges, is also a well-studied problem. We also remark on the relationship between the number of internal vertices in the maximum internal spanning tree and the number of edges in optimal path cover for the special graph classes mentioned above.

Chapter 3: Minimum Edge Total Dominating Set

In this chapter, we study the algorithmic and hardness results for the Min-ETDS problem. It is known that the decision version of the Min-ETDS problem is NP-complete for bipartite graphs and chordal graphs. We first prove that the problem is linear-time solvable for chain graphs, a subclass of bipartite graphs, and for two subclasses of chordal graphs, namely, split graphs and biconnected proper interval graphs. Next, we show that the problem is APX-complete for graphs with maximum degree 3 and propose an approximation algorithm for the problem in k -regular graphs, where $k \geq 4$. We discuss the complexity difference between the MINIMUM EDGE DOMINATING SET problem and MINIMUM EDGE TOTAL DOMINATING SET problem which seem to be closely related.

Chapter 4: Grundy (Double) Dominating Sequence

In this chapter, we give certain algorithmic and hardness results for the GDD and the GD2D problems in some restricted graph classes. First, we prove that the GDD problem is NP-complete for bipartite and co-bipartite graphs. Then we design a linear-time algorithm to compute a Grundy dominating sequence of a chain graph. For this purpose, we use the structure of a connected chain graph G so that we could characterize the structure of an optimal solution of G .

Next, we prove that the GD2D problem is NP-complete for bipartite, co-bipartite, and split graphs. Then we design a linear-time algorithm to compute a Grundy double dominating sequence of a chain graph and of a threshold graph. For designing the algorithm in chain graphs, we use the technique of dynamic programming in which we consider a slightly generalized version of the GD2D problem, defined only for chain graphs.

Chapter 5: Maximum Weighted Edge Biclique

In this chapter, we study the MWEB problem with some particular choices of the set from which the edge weights are being taken. The decision version of the MWEB problem is known to be NP-complete for bipartite graphs. We show that the decision version of the MWEB problem remains NP-complete even if the input graph is a complete bipartite graph. On the positive side, if the weight of each edge is a positive real number in the input graph G , then we show that the MWEB problem is $O(n^2)$ -time solvable for bipartite permutation graphs, and $O(m + n)$ -time solvable for chain graphs, which is a subclass of bipartite permutation graphs.

Chapter 6: Neighbor-Locating Coloring

In this chapter, we give some algorithms, hardness results, and bounds for computing a neighbor-locating chromatic number of a given graph G . We prove some bounds for general graphs and some improved bounds for proper interval graphs and chain graphs. For some special chain graphs, we compute the neighbor-locating chromatic number and find the corresponding coloring. One linear-time approximation algorithm is also presented for the problem. We remark on the complexity difference between the VERTEX COLORING problem and the NEIGHBOR-LOCATING COLORING problem. For a co-bipartite graph, we examine the neighbor-locating chromatic number.

Chapter 7: Conclusion and Future Directions

In this chapter, We first provide a summary of the results discussed in the thesis. Secondly, we present a list of prospective future directions and important unresolved questions.

Chapter 2

Maximum Internal Spanning Tree

2.1 Introduction

This chapter is devoted to the study of the MIST problem in graphs. The MIST problem aims to find a spanning tree of a given graph G with maximum number of internal vertices. Precisely, in this chapter, we study the complexity of the MIST problem in the following graph classes: block graphs, cactus graphs, cographs, bipartite permutation graphs, and chain graphs. We also discuss the relationship between the number of internal vertices in a MIST of the graph G and the number of edges in its optimal path cover. By optimal path cover, we mean a path cover with the maximum number of edges. Throughout this chapter, by $Opt(G)$, we mean the number of internal vertices in a MIST of G .

The well-known HAMILTONIAN PATH problem asks to determine whether there exists a Hamiltonian path in a given graph or not. The MIST problem generalizes the HAMILTONIAN PATH problem and the latter problem is NP-hard for bipartite and chordal graphs [54, 71]. Consequently, the Decide MIST problem is also NP-hard in general graphs as well as for bipartite and chordal graphs.

Note that we are working only with non-trivial connected graphs. So, a vertex that is not an internal vertex is a pendant vertex. Hence, finding a spanning tree of a given graph G with maximum number of internal vertices is equivalent of finding a spanning tree of G with minimum number of leaves. Due to this, researchers studied the dual problem to MIST, the MINIMUM LEAVES SPANNING TREE (MLST) problem which asks to find a spanning tree with the minimum number of leaves for a given graph. From the algorithmic point of view, both the MIST and the MLST problems are equivalent, but from the approximation aspect, status of both the problems differ. In 1992, Lu and Ravi proved the following theorem.

Theorem 2.1. [65] *The MLST problem cannot be approximated within any constant factor unless $P=NP$.*

Unlike MLST, several constant factor approximation algorithms have been proposed for the MIST problem in the literature. In 2003, Prieto et al. [80] gave a 2-approximation algorithm for the MIST problem whose running time was later improved by Salamon et al. in 2008 [83]. Salamon also gave approximation algorithms for claw-free and cubic graphs with approximation factors $\frac{3}{2}$ and $\frac{6}{5}$ respectively [83]. In 2009, Salamon [81] gave a $\frac{7}{4}$ -approximation algorithm for graphs with no pendant vertices and later, in 2015, Knauer et al. [52] showed that a simplified and faster version of Salamon's algorithm yields a $\frac{5}{3}$ -approximation algorithm even on general graphs. In 2014, Li et al. proposed a $\frac{3}{2}$ -approximation algorithm using a different approach for general undirected graphs and improved this ratio to $\frac{4}{3}$ for graphs without leaves [60]. Li et al. gave a $\frac{3}{2}$ -approximation algorithm for general graphs using depth-5 local search [58]. In 2018, Chen et al. presented a $\frac{17}{13}$ -approximation algorithm which is the best approximation factor till now [25]. Recently, Li et al. proved that the MIST problem is Max-SNP-hard [61].

For NP-hard problems, we generally look for some restricted graph classes in which the problem can be solved in polynomial time. In the case of the MIST problem, this search was unexplored for quite a long time. Recently, Li et al. proved that the class of interval graphs is one of the graph classes for which the Decide MIST problem is in the class P [57]. In this chapter, we show that for block graphs, cactus graphs, cographs, and bipartite permutation graphs, the MIST problem is efficiently solvable.

While designing such algorithms, it is often useful to find an optimal path cover of the graph. Recall that a path cover of a graph G is a spanning subgraph of G in which every component is a path. An optimal path cover of G is a path cover containing the maximum number of edges among all the path covers of G . In this chapter, P^* denotes an optimal path cover of the graph G under consideration. In 2018, researchers have proved a connection between $Opt(G)$ and $|E(P^*)|$.

Theorem 2.2. [59] *The number of internal vertices of a maximum internal spanning tree is less than the number of edges of an optimal path cover in a graph G , that is, $Opt(G) \leq |E(P^*)| - 1$, where P^* denotes an optimal path cover of G .*

Note that the vertices which are pendant in G itself will remain pendant in any MIST of G . Hence, we have the following lemma.

Lemma 2.3. *For a graph G , if v is a pendant vertex and $u = s(v)$ in G , then v remains a pendant vertex and u remains adjacent support vertex of v in any MIST of G .*

Suppose G is not a tree and $u \in V(G)$ is adjacent to $k \geq 2$ pendant vertices, say a_1, \dots, a_k . Let $G' = G \setminus \{a_2, \dots, a_k\}$. Then based on Lemma 2.3, the number of internal vertices in a MIST of G will be the same as the number of internal vertices in any MIST of G' . It is also easy to obtain a MIST of G from any MIST of G' . Hence, for this chapter, we assume that every vertex has at most one pendant vertex adjacent to it.

We denote a bipartite graph with the bipartition (X, Y) by $G = (X, Y, E)$. Below, we give another result regarding the number of pendant vertices in a spanning tree of a bipartite graph. Note that, if we have a number of internal vertices in a spanning tree of G from one partite set, then at least $a + 1$ vertices must be present in the neighborhood of these a vertices, which lie in the other partite set of the bipartite graph G .

Lemma 2.4. *Let $G = (X, Y, E)$ be a bipartite graph with $A \subseteq X$ and $B \subseteq Y$. If $N(A) = B$, then there are at least $\max\{0, |A| - |B| + 1\}$ pendant vertices from A in any spanning tree of G . Similarly, if $N(B) = A$, then there are at least $\max\{0, |B| - |A| + 1\}$ pendant vertices from B in any spanning tree of G .*

Now, we fix some terminologies. The set of pendant vertices in a graph G is denoted by $P(G)$. Let $I(G)$ denote the set of internal vertices in G , and $i(G) = |I(G)|$. For a set $A \subseteq V$ and a spanning tree T of G , we define $i_T(A) = |I(T) \cap A|$. Hence, for an empty set A , $i_T(A) = 0$.

The section-wise organization of this chapter is as follows: In Section 2.2, we present efficient algorithms to solve the MIST problem for all the graph classes mentioned at the beginning of this chapter. Section 2.3 describes the relationship between $Opt(G)$ and $|E(P^*)|$ for the same set of graph classes.

2.2 Polynomial-Time Algorithms

In this section, we design efficient algorithms to compute a maximum internal spanning tree for a given graph G when G is a block, cactus, cograph, or bipartite permutation graph.

2.2.1 Block and Cactus Graphs

Here, we discuss the results obtained for the MIST problem in block and cactus graphs in detail. First, we recall a few definitions.

A block of a graph G is a maximal connected subgraph with no cut vertices. The set of blocks of a graph is called the *block decomposition* of G and is denoted by $B(G)$. Let $B_0 \in B(G)$ and u, v be two vertices belonging to B_0 , then a path between u and v , which contains all the vertices of the block B_0 , is called a *spanning path between u and v in B_0* . We say a block B is *good* if there exist distinct vertices $u, v \in V(B)$ such that both u and v are cut vertices of G and B has a spanning path between u and v . A block is said to be *bad* otherwise. Let $\text{Bad}(G)$ denote the set of bad blocks of G .

A block graph is a graph in which every block is a clique. If a block graph G contains only one block then G is a complete graph. A block graph is said to be *nontrivial* if it contains at least two blocks. Note that a trivial block has a Hamiltonian path. Thus for the remainder of the subsection, we only consider nontrivial block graphs. Bad blocks of a block graph G have another characterization which we state as the Proposition 3.

Proposition 3. A block B of a nontrivial block graph G is bad if and only if it contains exactly one cut vertex of G .

Proof. A block containing exactly one cut vertex of G can not be a good block. So, assume B is a bad block of G and it contains at least 2 cut vertices of G . Then, these 2 vertices have a spanning path between them in B as $G[V(B)]$ is a clique. This implies that B is a good block, a contradiction. Hence, B contains exactly one cut vertex of G . \square

A graph G is a cactus graph if every block of G is either a cycle or an edge. If a cactus graph G contains only one block then G is either a cycle or an edge and in that case finding a

MIST of G is trivial. Again, a cactus graph is said to be *nontrivial* if it contains at least two blocks, and now we only consider nontrivial cactus graphs.

A block of a cactus graph G is called an *end block* of G if it contains exactly one cut vertex of G . Note that an end block of a cactus graph G is also a bad block of G . Bad blocks of a cactus graph G have another characterization which we state in the following Proposition.

Proposition 4. A block B of a nontrivial cactus graph G is bad if and only if B does not contain two adjacent cut vertices of G .

Proof. First, let B be a bad block of G . If B is an end block then it does not have two distinct cut vertices of G and so, there is nothing to prove. If B is not an end block, then it contains at least 2 cut vertices of G . Since B is a bad block and two adjacent vertices of a block of a cactus graph have a spanning path between them so, B does not contain two adjacent cut vertices of G . Conversely, let B be a block such that it does not contain two adjacent cut vertices of G . Then no two cut vertices of G have a spanning path between them in B . Hence, B is a bad block of G . \square

If B_i and B_j are two blocks of a block/cactus graph G and $V(B_i) \cap V(B_j) \neq \emptyset$, then $|V(B_i) \cap V(B_j)| = 1$ and the vertex $x \in V(B_i) \cap V(B_j)$ is a cut vertex of G . Below, we state two propositions that hold true for both block and cactus graphs.

Proposition 5. Let T be a MIST of a nontrivial block/cactus graph G . Then, T must have at least one leaf in every bad block of G .

Proof. Let B be a bad block of G . If B is an edge, then one vertex of B is itself pendant in G . So, we may assume that B is not an edge. Now, suppose that every vertex of block B is internal in T then the degree of each vertex of B is at least 2 in T . First, let G be a block graph. By Proposition 3, B has exactly one cut vertex of G , say u . Let $T' = T[V(B)]$. As T' is a forest, it must contain at least two leafs. As for any $x \in V(B) \setminus \{u\}$, $d_T(x) = d_{T'}(x)$, B must contain at least one leaf of T .

Now, let G be a cactus graph and $u \in V(B)$ be a cut vertex of G . Let x and y be neighbors of u in B . Since B is a bad block, by Proposition 4, x and y are non-cut vertices in G which implies that their degree is exactly 2 in G and edges xu, uy belong to T . Now, let $v \in V(B)$ be any non-cut vertex of G and let x' and y' be neighbors of v . Since $d_G(v)$ is 2 and it is an internal vertex in T , edges $x'v, vy'$ belong to T . So, we see that every edge of the block B of G belongs to T , a contradiction. Hence, T must have at least one leaf in every bad block. \square

Proposition 6. Let T be a MIST of a nontrivial block/cactus graph G . Then, $Opt(G) \leq n - |\text{Bad}(G)|$, where $Opt(G)$ denotes the number of internal vertices in T .

Proof. By Proposition 5, we have that $|P(T)| \geq |\text{Bad}(G)|$, where $P(T)$ denotes the set of leaves of T . So,

$$\begin{aligned} Opt(G) &= \text{number of internal vertices in a MIST of } G \\ &= n - \text{number of pendant vertices in a MIST of } G \\ &= n - |P(T)| \\ &\leq n - |\text{Bad}(G)| \end{aligned}$$

\square

The block decomposition of a graph G can be computed in $O(n)$ time using the following approach. Let b be a cut vertex of a block/cactus graph G and G_1, G_2, \dots, G_t be the connected components of the graph $G - b$. Let H_i denotes the subgraph $G[V(G_i) \cup \{b\}]$, for each $1 \leq i \leq t$. We call H_1, H_2, \dots, H_t the b -components of G . The block decomposition of a block/cactus graph can be found by recursively choosing a cut vertex b and computing the b -components.

Algorithm for Block and Cactus Graphs

Now, we first prove a theorem that relates the number of internal vertices in a MIST of a block/cactus graph G to the number of bad components of G . Then, we outline a linear-time algorithm to compute a MIST of G .

Theorem 2.5. *Let G be a graph with a nontrivial block decomposition such that each block has a spanning path with a cut vertex as an endpoint. Then G has a spanning tree T in which number of internal vertices is $n - |Bad(G)|$.*

Proof. Let l be the number of blocks in G and $B_i \in B(G)$ be an arbitrary block of G . If B_i is good, then let P_i be a spanning path between two cut vertices of B_i . If B_i is bad, we let P_i be a spanning path with a single cut vertex as an endpoint. Let $T = \bigcup_{i=1}^l P_i$. Note that T is a spanning tree of G . Furthermore, as any cut vertex of G cannot be a leaf of T , we have $i(T) = n - |Bad(G)|$. \square

The proof of Theorem 2.5 gives a simple algorithm for a block or cactus graph. First, find a block decomposition, this takes $O(n)$ time. Then for each block, B , determine if B is bad or not and find the corresponding path. This takes $O(|B|)$ time. In total, we have a linear-time algorithm. As both block and cactus graphs satisfy the hypothesis of Theorem 2.5, combining with Proposition 6 we have the following,

Corollary 2.6. *If G is a block or cactus graph, then $Opt(G) = n - |Bad(G)|$.*

2.2.2 Cographs

In this subsection, we discuss the MIST problem for cographs. The complement-reducible graphs or *cographs* have been discovered independently by several authors since the 1970s [84, 49]. In the literature, the cographs are also known as P_4 -free graphs, D^* -graphs, Hereditary Dacey graphs and 2-parity graphs. The class of cographs is defined recursively as follows:

- A graph containing a single vertex is a cograph;
- Complement of a cograph is also a cograph;
- Disjoint union of two cographs is also a cograph.

Cographs admit a rooted tree representation [56]. This tree is called a cotree of a cograph G , denoted $T(G)$. Cotree of a connected cograph satisfies the following properties:

1. Every internal vertex has at least two children.
2. Every internal vertex is labeled 0 or 1 with the root r receiving label 1, such that no two adjacent internal vertices receive the same label.
3. The leaves of $T(G)$ form a bijective correspondence with $V(G)$, such that $x, y \in V(G)$ are adjacent if and only if their least common ancestor of x and y in $T(G)$ has label 1.

Fig. 2.1 illustrates a cograph G along with its cotree $T(G)$.

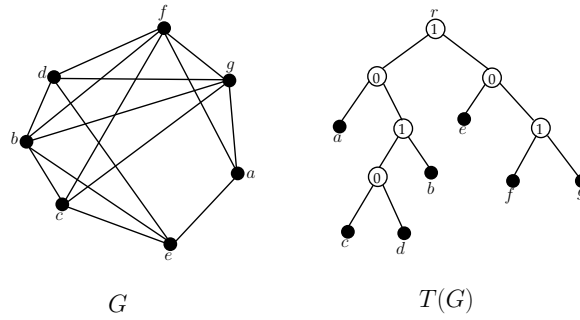


FIGURE 2.1: Illustrating a cograph and its cotree

Lin et al. [64], described a technique to preprocess a cotree $T(G)$ with root r such that it is a binary rooted tree possessing the property (3) of a cotree. Such a tree is called the binarized version of $T(G)$, denoted by $BT(G)$. Clearly, every internal vertex has exactly two children in $BT(G)$. The set of leaves of the left subtree (right subtree) of an interior vertex x of $BT(G)$ is denoted by $L(x_{left})$ ($L(x_{right})$). The tree $BT(G)$ also ensures that for every internal node with label 1, its left subtree contains at most as many leaves as the right subtree. Thus, we have that $BT(G)$ satisfies the following properties:

1. Every internal vertex has exactly two children.
2. The leaves of $BT(G)$ form a bijective correspondence with $V(G)$, such that $x, y \in V(G)$ are adjacent if and only if their least common ancestor of x and y in $BT(G)$ has label 1.
3. For all interior vertices x of $BT(G)$ assigned label 1, $|L(x_{left})| \leq |L(x_{right})|$.

Proposition 7. For any $x \in L(r_{left}), y \in L(r_{right})$, we have $xy \in E(G)$.

Proof. A leaf of $BT(G)$ represents a vertex of the graph G . Let $x \in L(r_{left}), y \in L(r_{right})$. As the least common ancestor of x and y is r and r has label 1, by property 3 of a cotree, $xy \in E(G)$. \square

A path cover P of a graph G is an optimal path cover if it has the maximum number of edges. Authors of [64] gave a linear-time algorithm to compute an optimal path cover of a cograph G . The optimal path cover P^* constructed in [64] is one of the following types:

- The path cover P^* contains a single path component which is a Hamiltonian path of G .
- The path cover P^* contains at least two path components. In this case, there exists exactly one path p in P^* which contains vertices from both the sets $L(r_{left})$ and $L(r_{right})$ and all other paths in P^* contain vertices from $L(r_{right})$ only. Fig. 2.2 illustrates this case.

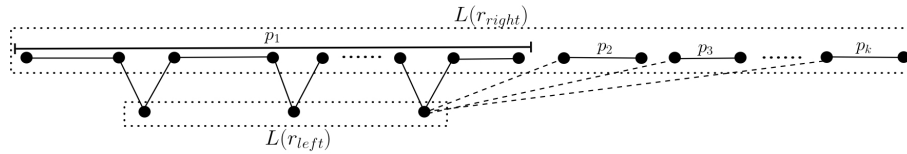


FIGURE 2.2: Optimal path cover of G containing more than 1 path components

Algorithm 1 uses the optimal path cover constructed from [64] to compute a MIST of a cograph G .

Note that by Theorem 2.2 we have $Opt(G) \leq |E(P^*)| - 1$ for an optimal path cover P^* . Below, we give a theorem that implies that Algorithm 1 also outputs a spanning tree which attains this upper bound.

Theorem 2.7. *Algorithm 1 outputs a spanning tree T of a cograph G such that, $i(T) = |E(P^*)| - 1$, where P^* is an optimal path cover of G . Hence, $Opt(G) = |E(P^*)| - 1$.*

Proof. Let $P^* = \{P_1, P_2, \dots, P_k\}$ be the optimal path cover computed in step 1 of Algorithm 1. If $|P^*| = 1$, then G has a Hamiltonian path and Algorithm 1 returns a Hamiltonian path. Now, suppose $|P^*| > 1$, then the path P_1 contains vertices from both sets $L(r_{left})$ and

Algorithm 1: Algorithm for finding a MIST of a cograph G **Input:** A cograph G and a cotree $T(G)$ of G .**Output:** A Maximum Internal Spanning Tree T of G .

```

1 Let  $P^* = \{P_1, P_2, \dots, P_k\}$  be the optimal path cover of  $G$  computed by the algorithm in
  [64];
2  $V(T) = V(G)$  and  $E(T) = E(P^*)$ ;
3 if  $k = 1$  then
4    $\quad$  return  $T$ ;
5 else
6    $\quad$  /*  $P_1$  is the path which contains vertices from both the sets  $L(r_{left})$  and  $L(r_{right})$  and
7     all other paths in  $P^*$  contain vertices from  $L(r_{right})$  only */
8    $\quad$  Let  $u \in (V(P_1) \cap L(r_{left}))$ ;
9    $\quad$  Let  $v_i$  be an end vertex of the path  $P_i$ , for  $2 \leq i \leq k$ ;
10   $\quad$   $E(T) = E(T) \cup \{uv_2, uv_3, \dots, uv_k\}$ ;
11   $\quad$  return  $T$ .

```

$L(r_{right})$ and $P_i \cap L(r_{left}) = \emptyset$ for all $i \geq 2$. Now, let $u \in V(P_1) \cap L(r_{left})$ such that u is not an end vertex of P_1 .

For each path in $P_i \in P^* \setminus \{P_1\}$, consider a pendent vertex v_i of the path. By Proposition 7, v_i and u are adjacent. Let $T = \bigcup_{i=1}^k P_i \cup \{v_i u : 2 \leq i \leq k\}$. These new edges connect one internal vertex with a pendant vertex of path of P^* . This is illustrated by the dash edges in Fig. 2.2. Note then the number of internal vertices of T is $|E(P^*)| - 1$, hence $i(T) = Opt(G) = |E(P^*)| - 1$ by Theorem 2.2. \square

Note that step 1 of Algorithm 1 can be performed in linear-time [64]. Furthermore, note that the construction of T in Theorem 2.7 is also linear-time. Therefore Algorithm 1 outputs a MIST of G in linear-time.

2.2.3 Bipartite Permutation Graphs

In this subsection, we discuss the MIST problem for bipartite permutation graphs. We first describe some properties of bipartite permutation graphs and after that, we present an efficient algorithm for the MIST problem in this graph class.

Let $G = (X, Y, E)$ be a connected bipartite permutation graph and $(<_X, <_Y)$ be a strong ordering of G , where $<_X = (x_1, x_2, \dots, x_{n_1})$ and $<_Y = (y_1, y_2, \dots, y_{n_2})$. For $u, v \in$

$V(G)$, we write $u <_X v$ if $u, v \in X$ and u appears before v in the strong ordering; we define $u <_Y v$ in a similar manner. We write $x_i < x_j$ (or, $y_i < y_j$) when $i < j$. For vertices u, v of G , $u \leq v$ denotes either $u <_X v$, $u <_Y v$, or $u = v$ holds.

Since each vertex of G satisfies the adjacency property (see, lemma 1.4), the neighborhood of any vertex consists of consecutive vertices in the strong ordering. We define the *first neighbor* of a vertex as the vertex with minimum index in its neighborhood and the *last neighbor* of a vertex as the vertex with maximum index in its neighborhood. We notate the first and last neighbors of a vertex u as $f(u)$ and $l(u)$ respectively. Combining the above statements for a bipartite permutation graph G with its strong ordering $(<_X, <_Y)$, G has the following properties [55]:

1. For any vertex of G , vertices in its neighborhood are consecutive with respect to the ordering $<_X$ or $<_Y$.
2. If $u < v$ then $f(u) \leq f(v)$ and $l(u) \leq l(v)$, for each pair of vertices $u, v \in V(G)$.

Now, we define some terminology which we require for the remainder of this subsection. A vertex $x_i \in X, (1 \leq i \leq n_1)$ with $l(x_i) = y_j$ is of *type 1* if $j \geq i$. A vertex $y_i \in Y, (1 \leq i \leq n_2)$ with $l(y_i) = x_j$ is of *type 1* if $j \geq i + 1$. Similarly, a vertex $x_i \in X, (1 \leq i \leq n_1)$ with $l(x_i) = y_j$ is of *type 2* if $j \geq i + 1$ and a vertex $y_i \in Y, (1 \leq i \leq n_2)$ with $l(y_i) = x_j$ is of *type 2* if $j \geq i$. Note that a type 2 vertex $x \in X$ is also a type 1 vertex but the converse may not be true. Furthermore, a type 1 vertex $y \in Y$ is also a type 2 vertex. Characterizing the vertices in this way is an important distinction for our algorithm. We now prove two important lemmas which will be used to prove the correctness of Algorithm 2.

Lemma 2.8. *Let $X' = \{x_1, x_2, \dots, x_k, x_{k+1}\} \subseteq X$, $Y' = \{y_1, y_2, \dots, y_k\} \subseteq Y$. Furthermore, suppose each vertex of X' and Y' is of type 1 except x_{k+1} , $l(x_{k+1}) = y_k$ and $N(X') = Y'$. Then there exists a MIST T of G , in which x_1 and x_{k+1} are pendant. Moreover; if $X \setminus X' \neq \emptyset$, then the support vertex of x_{k+1} is of degree at least 3 in T .*

Proof. We first show $x_i y_i, y_i x_{i+1} \in E(G)$ for all $1 \leq i \leq k$. Suppose there exists $1 \leq i \leq k$ such that $x_i y_i \notin E(G)$. Let $l(x_i) = y_j$ and $l(y_i) = x_l$. As both x_i and y_i are type 1, we have

$y_j \geq y_i$ and $x_l > x_i$. As $(<_X, <_Y)$ is a strong ordering, we have $x_i y_i \in E(G)$, a contradiction. Thus we may assume $x_i y_i \in E(G)$. Furthermore, as y_i is of type 1 and $x_i y_i \in E(G)$, we have $y_i x_{i+1} \in E(G)$. Hence, we may conclude that $x_i y_i, y_i x_{i+1} \in E(G)$ for all i , $1 \leq i \leq k$.

First, assume that $X = X'$. Note that as $(<_X, <_Y)$ is a strong ordering of G , we have for all $x \in X$, $l(x) \leq l(x_{k+1}) = y_k$. Since G is connected, we have, $Y = Y'$ as well. This implies that G has a Hamiltonian path $x_1 y_1 x_2 \dots x_k y_k x_{k+1}$ which is a MIST.

Now, we may assume that $X \setminus X' \neq \emptyset$. Let T^* be a MIST of G . If x_1 and x_{k+1} are pendant in T^* and degree of $S(x_{k+1})$ is at least 3 in T^* , then we are done. Suppose otherwise. In that case, we modify T^* in the following way. We first remove all edges of T^* incident with the vertices of X' and then add edges $x_1 y_1, y_1 x_2, x_2 y_2, \dots, x_k y_k$ and $y_k x_{k+1}$ to obtain a new graph T . Note that as $N(X') = Y'$, T is connected.

First suppose that T contains no cycle. Note that T is a spanning tree of G . If $d_T(y_k) = 2$, then as $N(X') = Y'$ we can choose an edge vy_i ($i < k$) in T such that $v \in X \setminus X'$. Since the strong ordering $(<_X, <_Y)$ of the vertices of G satisfies property 2, we have $vy_k \in E(G)$. So we can further modify T by removing the edge vy_i and replacing with the edge vy_k . Note that $i_{T^*}(X') \leq k - 1$. To see this, if $i_{T^*}(X') > k - 1$ then the subgraph $T^*[X' \cup Y']$ has at least $2k + 1$ edges and exactly $2k + 1$ vertices. This implies that $T^*[X' \cup Y']$ contains a cycle, a contradiction as T^* is a tree. We see that

$$\begin{aligned} i(T^*) &= i_{T^*}(X') + i_{T^*}(X \setminus X') + i_{T^*}(Y') + i_{T^*}(Y \setminus Y') \\ &\leq (k - 1) + i_{T^*}(X \setminus X') + k + i_{T^*}(Y \setminus Y') = i(T). \end{aligned}$$

So, we have $i(T^*) \leq i(T)$. Since T is a spanning tree and T^* is a MIST of G , we have that T is also a MIST. Thus, we obtain our desired MIST in which x_1 and x_{k+1} are pendant and the support vertex of x_{k+1} is of degree at least 3.

Now, suppose T contains a cycle C . Then there exists a pair of vertices $y_i, y_j \in Y'$ ($i < j$) such that the path between y_i and y_j in T^* contains no vertex of X' . Let P denotes this path. Suppose x_p is the vertex adjacent to y_i in P and x_q is the vertex adjacent to y_j in P . Now, if $d_T(x_p) \geq 3$ or $d_T(x_p) = d_T(x_q) = 2$, we modify T by removing the edge $y_i x_p$.

Otherwise, if $d_T(x_q) \geq 3$, we modify T by removing the edge $y_j x_q$. In this way, we removed the cycle C . We repeat the same modification until there is no cycle in T . Thus, we end up with a spanning tree T .

In this process of removing cycles, we made some vertices of $X \setminus X'$ pendant in T which were internal in T^* . Let a_0 be the number of vertices of $X \setminus X'$ which were internal in T^* and they are pendant in T . We denote the set of these a_0 vertices by K . For each vertex of K , there exists a pair of vertices $y_i, y_j \in Y'$ ($i < j$) such that the path between y_i and y_j in T^* contains no vertex of X' . Note that we have such a path for each vertex of K . Let \mathcal{K} be the set of all these paths. We claim that $i_{T^*}(X') \leq k - a_0 - 1$. To see this, assume $i_{T^*}(X') \geq k - a_0$. Then, we define $A = \{V(P) \cap (X \setminus X') : P \in \mathcal{K}\}$ and $B = \{V(P) \cap (Y \setminus Y') : P \in \mathcal{K}\}$. Now, consider the subgraph H of T^* induced on $X' \cup Y' \cup A \cup B$. We see that $|V(H)| = 2k + 1 + |A| + |B|$. Recall that each vertex of K correspond to a pair of vertices of Y' that has a path between them with no vertex of X' in the tree T^* . Note that each vertex of K correspond to a unique path. Let p_1, p_2, \dots, p_{a_0} be these a_0 paths. For each $i \in [a_0]$, we denote the number of intermediate vertices in the path p_i by $IV(p_i)$. Now, $|E(H)| = |E_{X'Y'}^*| + |E_{Y'A}^*| + |E_{AB}^*|$, where $E_{X'Y'}^*$ denotes the set of edges having one end point in X' and another end point in Y' in the tree T^* . Similarly, $E_{Y'A}^*$ denotes the set of edges having one end point in Y' and another end point in A in the tree T^* . Again, E_{AB}^* denotes the set of edges having one end point in A and another end point in B in the tree T^* . We have,

$$\begin{aligned}
|E(H)| &= |E_{X'Y'}^*| + |E_{Y'A}^*| + |E_{AB}^*| \\
&\geq |E_{X'Y'}^*| + |E(p_1 \cup p_2 \cup \dots \cup p_{a_0})| \\
&= |E_{X'Y'}^*| + |E(p_1)| + |E(p_2) \setminus E(p_1)| + \dots + |E(p_{a_0}) \setminus E(\cup_{j=1}^{a_0-1} p_j)| \\
&\geq |E_{X'Y'}^*| + (|IV(p_1)| + 1) + (|IV(p_2) \setminus IV(p_1)| + 1) + \dots + (|IV(p_{a_0}) \setminus IV(\cup_{j=1}^{a_0-1} p_j)| + 1) \\
&= |E_{X'Y'}^*| + IV(p_1 \cup p_2 \cup \dots \cup p_{a_0}) + a_0 \\
&\geq |E_{X'Y'}^*| + |A| + |B| + a_0 \\
&\geq 2(k - a_0) + (a_0 + 1) + |A| + |B| + a_0 \\
&= 2k + 1 - a_0 + |A| + |B| + a_0 \\
&= 2k + 1 + |A| + |B|.
\end{aligned}$$

Thus, $|E(H)| \geq 2k + 1 + |A| + |B|$. This means that $|E(H)| \geq |V(H)|$ implying that H contains a cycle, a contradiction on the fact that T^* is a tree. Hence, $i_{T^*}(X') \leq k - a_0 - 1$. It follows,

$$\begin{aligned} i(T^*) &= i_{T^*}(X') + i_{T^*}(X \setminus X') + i_{T^*}(Y') + i_{T^*}(Y \setminus Y') \\ &\leq (k - 1 - a_0) + i_{T^*}(X \setminus X') + k + i_{T^*}(Y \setminus Y') \\ &= (k - 1) + (i_{T^*}(X \setminus X') - a_0) + k + i_{T^*}(Y \setminus Y') = i(T). \end{aligned}$$

So, we have $i(T^*) \leq i(T)$ which implies that T is also a MIST. If $d_T(y_k) = 2$, then we can choose an edge vy_i ($i < k$) in T , such that $v \in X \setminus X'$. Since the strong ordering $(<_X, <_Y)$ satisfies property 2, we have $vy_k \in E(G)$. So we update the tree T by removing the edge vy_i and adding the edge vy_k . Thus, we obtain a MIST T in which x_1 and x_{k+1} are pendant and support vertex of x_{k+1} , that is, y_k is of degree at least 3. \square

Lemma 2.9. *Let $X' = \{x_1, x_2, \dots, x_k\} \subseteq X$, $Y' = \{y_1, y_2, \dots, y_k\} \subseteq Y$. Furthermore, suppose each vertex of X' and Y' is of type 1 except y_k , $l(y_k) = x_k$ and $N(Y') = X'$.*

- (a) *If $x_i y_{i+1} \in E(G)$ for all $1 \leq i \leq (k - 1)$, then there exists a MIST T of G , in which y_1 is pendant.*
- (b) *If there exists $1 \leq t \leq (k - 1)$ such that $x_t y_{t+1} \notin E(G)$, then there exists a MIST T of G , in which x_1 and y_k are pendant. Moreover; if $Y \setminus Y' \neq \emptyset$, then support vertex of y_k is of degree at least 3 in T .*

Proof. We first argue that $x_i y_i, y_i x_{i+1} \in E(G)$ for $1 \leq i \leq k - 1$. First assume for some i , $x_i y_i \notin E(G)$. As both x_i and y_i are type 1, we have $x_i < l(y_i)$ and $y_i < l(x_i)$. As $(<_X, <_Y)$ is a strong ordering, we have $x_i y_i \in E(G)$, a contradiction. Furthermore, as y_i is of type 1, we have $y_i x_{i+1} \in E(G)$. Hence, we may conclude that $x_i y_i, y_i x_{i+1} \in E(G)$ for all i , $1 \leq i \leq k - 1$.

First assume that $Y' = Y$. As $N(Y') = X'$, and G is connected we have, $X = X'$ as well. Note that if $x_i y_{i+1} \in E(G)$ for all $1 \leq i \leq (k - 1)$, then $y_1 x_1 \dots x_{k-1} y_k x_k$ is a

Hamiltonian path. Otherwise, if there exists $1 \leq t \leq (k-1)$ such that $x_t y_{t+1} \notin E(G)$, then $x_1 y_1 x_2 y_2 \dots y_{k-1} x_k y_k$ gives the desired Hamiltonian path.

Now, assume that $Y \setminus Y' \neq \emptyset$. We will first prove part (a). So, we also assume that $x_i y_{i+1} \in E(G) \forall 1 \leq i \leq k-1$. Let T^* be a MIST of G and suppose y_1 is not pendant in T^* . Let T be the graph obtained from T^* where we remove all edges incident to the vertices of Y' and add edges $y_1 x_1, x_1 y_2, y_2 x_2, \dots, x_{k-1} y_k$ and $y_k x_k$. Note that y_1 is pendant in T .

First, suppose that T contains no cycles. In that case, T is a spanning tree of G . If $d_T(x_k) = 1$, we modify T to make $d_T(x_k) \geq 2$. Let $v \in Y \setminus Y'$ such that $vx_i \in E(T)$ where $i < k$. As the strong ordering of the vertices of G satisfies property 2, we have $vx_k \in E(G)$ as well. So we modify T by removing the edge vx_i and adding the edge vx_k . Note that T still remains a spanning tree of G . Next, we claim that $i(T^*) \leq i(T)$.

$$\begin{aligned} i(T^*) &= i_{T^*}(X') + i_{T^*}(X \setminus X') + i_{T^*}(Y') + i_{T^*}(Y \setminus Y') \\ &\leq k + i_{T^*}(X \setminus X') + (k-1) + i_{T^*}(Y \setminus Y') = i(T). \end{aligned}$$

So, we have that $i(T^*) \leq i(T)$. Since T is a spanning tree and T^* is a MIST of G , T is also a MIST of G .

Next, suppose T is not a tree, that is, T contains a cycle. We now modify T to remove the cycles. Let C be a cycle of T . Then there exists a pair of vertices $x_i, x_j \in X'$ ($i < j$) such that the path between x_i and x_j in T^* contains no vertex of Y' . Let P denotes this path. Suppose y_p is the vertex adjacent to x_i in P and y_q is the vertex adjacent to x_j in P . Now, if $d_T(y_p) \geq 3$ or $d_T(y_p) = d_T(y_q) = 2$, we modify T by removing the edge $x_i y_p$. Otherwise, if $d_T(y_q) \geq 3$, we modify T by removing the edge $x_j y_q$. In this way, we removed the cycle C . We repeat the same modification until there is no cycle in T . Thus, we end up with a spanning tree T .

In this process of removing cycles, we made some vertices of $Y \setminus Y'$ pendant in T which were internal in T^* . Let a_0 be the number of vertices of $Y \setminus Y'$ which were internal in T^* and they are pendant in T . We denote the set of these a_0 vertices by K . For each vertex of K , there exists a pair of vertices $x_i, x_j \in X'$ ($i < j$) such that the path between

x_i and x_j in T^* contains no vertex of Y' . Note that we have such a path for each vertex of K . Let \mathcal{K} be the set of all these paths. We claim that $i_{T^*}(Y') \leq k - a_0 - 1$. To see this, assume $i_{T^*}(Y') \geq k - a_0$. Then, we define $A = \{V(P) \cap (X \setminus X') : P \in \mathcal{K}\}$ and $B = \{V(P) \cap (Y \setminus Y') : P \in \mathcal{K}\}$. Now, consider the subgraph H of T^* induced on $X' \cup Y' \cup A \cup B$. We see that $|V(H)| = 2k + |A| + |B|$. Recall that each vertex of K correspond to a pair of vertices of X' that has a path between them with no vertex of Y' in the tree T^* . Note that each vertex of K correspond to a unique path. Let p_1, p_2, \dots, p_{a_0} be these a_0 paths. For each $i \in [a_0]$, we denote the number of intermediate vertices in the path p_i by $IV(p_i)$. Now, $|E(H)| = |E_{X'Y'}^*| + |E_{X'B}^*| + |E_{AB}^*|$, where $E_{X'Y'}^*$ denotes the set of edges having one end point in X' and another end point in Y' in the tree T^* . Similarly, $E_{X'B}^*$ denotes the set of edges having one end point in X' and another end point in B in the tree T^* . Again, E_{AB}^* denotes the set of edges having one end point in A and another end point in B in the tree T^* . We have,

$$\begin{aligned}
|E(H)| &= |E_{X'Y'}^*| + |E_{X'B}^*| + |E_{AB}^*| \\
&\geq |E_{X'Y'}^*| + |E(p_1 \cup p_2 \cup \dots \cup p_{a_0})| \\
&= |E_{X'Y'}^*| + |E(p_1)| + |E(p_2) \setminus E(p_1)| + \dots + |E(p_{a_0}) \setminus E(\cup_{j=1}^{a_0-1} p_j)| \\
&\geq |E_{X'Y'}^*| + (|IV(p_1)| + 1) + (|IV(p_2) \setminus IV(p_1)| + 1) + \dots + (|IV(p_{a_0}) \setminus IV(\cup_{j=1}^{a_0-1} p_j)| + 1) \\
&= |E_{X'Y'}^*| + IV(p_1 \cup p_2 \cup \dots \cup p_{a_0}) + a_0 \\
&\geq |E_{X'Y'}^*| + |A| + |B| + a_0 \\
&\geq 2(k - a_0) + a_0 + |A| + |B| + a_0 \\
&= 2k - a_0 + |A| + |B| + a_0 \\
&= 2k + |A| + |B|.
\end{aligned}$$

Thus, $|E(H)| \geq 2k + |A| + |B|$. This means that $|E(H)| \geq |V(H)|$ implying that H contains a cycle, a contradiction on the fact that T^* is a tree. Hence, $i_{T^*}(Y') \leq k - a_0 - 1$. It follows,

$$\begin{aligned}
i(T^*) &= i_{T^*}(X') + i_{T^*}(X \setminus X') + i_{T^*}(Y') + i_{T^*}(Y \setminus Y') \\
&\leq k + i_{T^*}(X \setminus X') + (k - a_0 - 1) + i_{T^*}(Y \setminus Y') \\
&\leq k + i_{T^*}(X \setminus X') + (k - 1) + (i_{T^*}(Y \setminus Y') - a_0) = i(T)
\end{aligned}$$

Again, we have that $i(T^*) \leq i(T)$ which implies that T is also a MIST. Hence, part (a) holds.

Next, we prove part (b). Let T^* be a MIST of G . If x_1 and y_k are pendant in T^* and degree of $S(y_k)$ is at least 3 in T^* , then we are done, so assume otherwise. In that case, let T be the graph obtained by modifying T^* where we remove all edges incident on the vertices of Y' and add edges $x_1y_1, y_1x_2, x_2y_2, \dots, y_{k-1}x_k$ and x_ky_k .

First suppose that T contains no cycle, then T is a spanning tree of G . If $d_T(x_k) = 2$, we further modify T to make $d_T(x_k) \geq 3$. As $Y \setminus Y' \neq \emptyset$ and $N(Y') = X'$, there exists an edge $vx_i \in E(T)$ where $i < k$ and $v \in Y \setminus Y'$. Since the strong ordering of the vertices of G satisfies property 2, we have $vx_k \in E(G)$. Thus we further modify T by removing vx_i and adding the edge vx_k . As we assumed that there exists a $t, 1 \leq t \leq (k-1)$ such that $x_t y_{t+1} \notin E(G)$, we have $N(\{x_1, x_2, \dots, x_t\}) = \{y_1, y_2, \dots, y_t\}$. Let $X'' = \{x_1, x_2, \dots, x_t\}$ and note $N(X'') = Y'' = \{y_1, y_2, \dots, y_t\}$. By Lemma 2.4, we see that for any spanning tree of G , X'' contains at least one pendant vertex. So, $i_{T^*}(X') \leq (k-1)$. We see that

$$\begin{aligned} i(T^*) &= i_{T^*}(X') + i_{T^*}(X \setminus X') + i_{T^*}(Y') + i_{T^*}(Y \setminus Y') \\ &\leq (k-1) + i_{T^*}(X \setminus X') + (k-1) + i_{T^*}(Y \setminus Y') = i(T). \end{aligned}$$

Again, we have $i(T^*) \leq i(T)$ which implies that T is a MIST. Thus, we obtained a MIST T in which x_1 and y_k are pendant and support vertex of y_k is of degree at least 3.

Now, suppose that T contains a cycle. We now modify T to be a spanning tree by removing cycles. Let C be a cycle contained in T . Then there exists a pair of vertices $x_i, x_j \in X'$ ($i < j$) such that the path between x_i and x_j in T^* contains no vertex of Y' . Let P denotes this path. Suppose y_p is the vertex adjacent to x_i in P and y_q is the vertex adjacent to x_j in P . Now, if $d_T(y_p) \geq 3$ or $d_T(y_p) = d_T(y_q) = 2$, we modify T by removing the edge $x_i y_p$. Otherwise, if $d_T(y_q) \geq 3$, we modify T by removing the edge $x_j y_q$. In this way, we removed the cycle C . We repeat the same modification until there is no cycle in T . Thus, we end up with a spanning tree T .

In this process of removing cycles, we made some vertices of $Y \setminus Y'$ pendant in T which were internal in T^* . Let a_0 be the number of vertices of $Y \setminus Y'$ which were internal

in T^* and they are pendant in T . We denote the set of these a_0 vertices by K . For each vertex of K , there exists a pair of vertices $x_i, x_j \in X'$ ($i < j$) such that the path between x_i and x_j in T^* contains no vertex of Y' . Note that we have such a path for each vertex of K . Let \mathcal{K} be the set of all these paths. We claim that $i_{T^*}(Y') \leq k - a_0 - 1$. To see this, assume $i_{T^*}(Y') \geq k - a_0$. Then, we define $A = \{V(P) \cap (X \setminus X') : P \in \mathcal{K}\}$ and $B = \{V(P) \cap (Y \setminus Y') : P \in \mathcal{K}\}$. Now, consider the subgraph H of T^* induced on $X' \cup Y' \cup A \cup B$. We see that $|V(H)| = 2k + |A| + |B|$. Recall that each vertex of K correspond to a pair of vertices of X' that has a path between them with no vertex of Y' in the tree T^* . Note that each vertex of K correspond to a unique path. Let p_1, p_2, \dots, p_{a_0} be these a_0 paths. For each $i \in [a_0]$, we denote the number of intermediate vertices in the path p_i by $IV(p_i)$. Now, $|E(H)| = |E_{X'Y'}^*| + |E_{X'B}^*| + |E_{AB}^*|$, where $E_{X'Y'}^*$ denotes the set of edges having one end point in X' and another end point in Y' in the tree T^* . Similarly, $E_{X'B}^*$ denotes the set of edges having one end point in X' and another end point in B in the tree T^* . Again, E_{AB}^* denotes the set of edges having one end point in A and another end point in B in the tree T^* . We have,

$$\begin{aligned}
|E(H)| &= |E_{X'Y'}^*| + |E_{X'B}^*| + |E_{AB}^*| \\
&\geq |E_{X'Y'}^*| + |E(p_1 \cup p_2 \cup \dots \cup p_{a_0})| \\
&= |E_{X'Y'}^*| + |E(p_1)| + |E(p_2) \setminus E(p_1)| + \dots + |E(p_{a_0}) \setminus E(\cup_{j=1}^{a_0-1} p_j)| \\
&\geq |E_{X'Y'}^*| + (|IV(p_1)| + 1) + (|IV(p_2) \setminus IV(p_1)| + 1) + \dots + (|IV(p_{a_0}) \setminus IV(\cup_{j=1}^{a_0-1} p_j)| + 1) \\
&= |E_{X'Y'}^*| + IV(p_1 \cup p_2 \cup \dots \cup p_{a_0}) + a_0 \\
&\geq |E_{X'Y'}^*| + |A| + |B| + a_0 \\
&\geq 2(k - a_0) + a_0 + |A| + |B| + a_0 \\
&= 2k - a_0 + |A| + |B| + a_0 \\
&= 2k + |A| + |B|.
\end{aligned}$$

Thus, $|E(H)| \geq 2k + |A| + |B|$. This means that $|E(H)| \geq |V(H)|$ implying that H contains a cycle, a contradiction on the fact that T^* is a tree. Hence, $i_{T^*}(Y') \leq k - a_0 - 1$.

It follows,

$$\begin{aligned}
 i(T^*) &= i_{T^*}(X') + i_{T^*}(X \setminus X') + i_{T^*}(Y') + i_{T^*}(Y \setminus Y') \\
 &\leq (k-1) + i_{T^*}(X \setminus X') + (k - a_0 - 1) + i_{T^*}(Y \setminus Y') \\
 &\leq (k-1) + i_{T^*}(X \setminus X') + (k-1) + (i_{T^*}(Y \setminus Y') - a_0) = i(T)
 \end{aligned}$$

This implies that T is also a MIST. Thus, we obtained a MIST T in which x_1 and y_k are pendant and support vertex of y_k is of degree at least 3. \square

We state similar results when the vertices are of type 2. By symmetry, the proofs of Lemmas 2.10 and 2.11 follow from Lemmas 2.8 and 2.9.

Lemma 2.10. *Let $X' = \{x_1, x_2, \dots, x_k\} \subseteq X$, $Y' = \{y_1, y_2, \dots, y_k, y_{k+1}\} \subseteq Y$. Furthermore, suppose each vertex of X' and Y' is of type 2 except y_{k+1} , $l(y_{k+1}) = x_k$ and $N(Y') = X'$. Then there exists a MIST T of G , in which y_1 and y_{k+1} are pendant. Moreover; if $Y \setminus Y' \neq \emptyset$, then support vertex of y_{k+1} is of degree at least 3 in T .*

Lemma 2.11. *Let $X' = \{x_1, x_2, \dots, x_k\} \subseteq X$, $Y' = \{y_1, y_2, \dots, y_k\} \subseteq Y$. Furthermore, suppose each vertex of X' and Y' is of type 2 except x_k , $l(x_k) = y_k$ and $N(X') = Y'$.*

- (a) *If $y_i x_{i+1} \in E(G) \forall 1 \leq i \leq (k-1)$, then there exists a MIST T of G , in which x_1 is pendant.*
- (b) *If $\exists 1 \leq t \leq (k-1)$ such that $y_t x_{t+1} \notin E(G)$, then there exists a MIST T of G , in which y_1 and x_k are pendant. Moreover; if $X \setminus X' \neq \emptyset$, then support vertex of x_k is of degree at least 3 in T .*

Algorithm for Bipartite Permutation Graphs

Now, we propose an algorithm to find a MIST of G based on the Lemmas 2.8, 2.9, 2.10 and 2.11. In our algorithm, we first find a vertex u such that it is a pendant vertex in some MIST T of G and the degree of support vertex of u in T is at least 3. Now, if we remove u from G and call the remaining graph G' , then we see that the number of internal vertices in a MIST of G is same as the number of internal vertices in a MIST of G' . Note that we

can easily construct a MIST of G from a MIST of G' by adding the pendant vertex u to the corresponding support vertex. So, after finding the vertex u , the problem is reduced to finding MIST of $G \setminus \{u\}$, say G' . We continue doing the same until no such vertex u exists and then the resulting graph has a Hamiltonian path.

In our algorithm, we visit the vertices alternatively from the partitions X and Y . We consider two special orderings $(x_1, y_1, x_2, y_2, \dots)$ and $(y_1, x_1, y_2, x_2, \dots)$ of $V(G)$ which we call α_0 and β_0 , respectively. Below, we describe the method to find a vertex u which is a pendant vertex in some MIST T of G and $d_T(s(u))$ is at least 3.

We first visit the vertices of G in the ordering α_0 and search for the first vertex, which is not of type 1. Let u be such a vertex. If $u \in X$ or $u \in Y$ and the conditions of part (b) of Lemma 2.9 are satisfied, then there exists a MIST T of G in which u is a pendant vertex and the degree of support vertex of u in T is at least 3. So, we remove u from G and find a MIST T' of $G \setminus \{u\}$. Later, we obtain a MIST of G by adding u to T' . But, if $u \in Y$, say $u = y_k$ and conditions of part (a) of Lemma 2.9 are satisfied, then there exists a MIST T of G in which y_1 is a pendant vertex. In this case, we start visiting the vertices of G in the ordering β_0 , starting from y_1 . At this step, we do not maintain any information from α_0 search.

Now, let u be the first vertex not of type 2 in the ordering β_0 . If $u \in Y$ or $u \in X$ and the conditions of part (b) of Lemma 2.11 are satisfied, then there exists a MIST T of G in which u is a pendant vertex and the degree of support vertex of u in T is at least 3. So, we remove u from G and find a MIST T' of $G \setminus \{u\}$. Later, we obtain a MIST of G by adding u to T' . Here, if $u \in X$ and conditions of part (a) of Lemma 2.11 are satisfied, then there exists a MIST T of G in which x_1 is a pendant vertex. But, we have already explored this possibility while visiting the vertices of G in the ordering α_0 . So, we do not get such a vertex u . To see this, suppose that we get such a vertex u . Then, $u = x_t$ for some t , where $t > k$. Now, part (a) of Lemma 2.11 tells that $y_i x_{i+1} \in E(G)$ for all $1 \leq i \leq (t-1)$ implying that $y_k x_{k+1} \in E(G)$. But, while visiting the vertices in the ordering α_0 , we got a vertex y_k satisfying $l(y_k) = x_k$, so $y_k x_{k+1} \notin E(G)$, a contradiction.

The detailed procedure for computing a MIST of a bipartite permutation graph is presented in Algorithm 2. Algorithm 2 either finds a vertex which is not of type 1 or a vertex

which is not of type 2. When such a vertex u is found, we call u as an *encountered* vertex. All the encountered vertices are found while executing the steps written in lines 4, 11, 17, 22, 31 or 39 of Algorithm 2. We see that the algorithm returns a spanning tree T of G . Before proving the correctness of the Algorithm 2, we state a necessary lemma.

Lemma 2.12. *Let G be the input bipartite permutation graph for the Algorithm 2 and a_1 denotes the first encountered vertex in either the α_0 or β_0 search. Suppose that T is the spanning tree of G returned by Algorithm 2. Let $X_1 \subseteq X$ be the set of vertices which are visited from X side till a_1 and $Y_1 \subseteq Y$ be the set of vertices which are visited from Y side till a_1 . Then there exists a MIST T^* of G such that $E(T^*[X_1 \cup Y_1]) = E(T[X_1 \cup Y_1])$.*

Proof. We have four cases to consider.

Case 1: $a_1 \in X$ and it is not of type 1. Then the vertex a_1 was found when $\text{flag} = 1$ in Algorithm 2, that is, when searching for the first vertex not of type 1. Let $a_1 = x_{k+1}$ for some k . Then the sets $X' = \{x_1, x_2, \dots, x_k, x_{k+1}\} \subseteq X$, $Y' = \{y_1, y_2, \dots, y_k\} \subseteq Y$ satisfy the hypothesis of Lemma 2.8. Thus by Lemma 2.8, there exists a MIST T^* of G such that $E(T^*[X_1 \cup Y_1]) = \{x_1y_1, y_1x_2, x_2y_2, \dots, x_ky_k, y_kx_{k+1}\}$. In particular, $E(T^*[X_1 \cup Y_1]) = E(T[X_1 \cup Y_1])$.

Case 2: $a_1 \in Y$ and it is not of type 1. Then the vertex a_1 was also found when $\text{flag} = 1$ in the algorithm. Let $a_1 = y_k$ for some k . Then the sets $X' = \{x_1, x_2, \dots, x_k\} \subseteq X$, $Y' = \{y_1, y_2, \dots, y_k\} \subseteq Y$ satisfy the hypothesis of part (b) of Lemma 2.9. Thus by Lemma 2.9, there exists a MIST T^* of G such that $E(T^*[X_1 \cup Y_1]) = \{x_1y_1, y_1x_2, x_2y_2, \dots, x_ky_k\} = E(T[X_1 \cup Y_1])$.

By symmetry, the other two cases ($a_1 \in X$ and it is not of type 2; $a_1 \in Y$ and it is not of type 2) follow from Lemmas 2.10 and 2.11. Thus there exists a MIST T^* of G such that $E(T^*[X_1 \cup Y_1]) = E(T[X_1 \cup Y_1])$ in all cases. \square

Now, we prove the correctness of Algorithm 2.

Theorem 2.13. *Algorithm 2 returns a maximum internal spanning tree of G .*

Algorithm 2: Algorithm for finding a MIST of a bipartite permutation graph G**Input:** A bipartite permutation graph G and a strong ordering $(\prec_X, \prec_Y) = (x_1, x_2, \dots, x_{n_1}, y_1, y_2, \dots, y_{n_2})$ of $V(G)$. **Output:** A MIST T of G .

```

1   $V(T) = X \cup Y, E(T) = \emptyset, t = 0; flag = 1;$ 
2   $\alpha_0 = (x_1, y_1, x_2, y_2, \dots)$  and  $\beta_0 = (y_1, x_1, y_2, x_2, \dots);$ 
3  Visit the vertices of  $V(G)$  in the ordering  $\alpha_0$ ;
4  Let  $u$  be the first vertex with minimum index in the ordering  $\alpha_0$  which is not of type 1;
5  while  $flag == 1$  do
6      if  $u \in X$  then
7          Let  $u = x_{k+1}$  for some  $k$ ;
8          if  $k + 1 \neq n_1$  then
9               $t = t + 1$ ; rename  $x_{k+1}$  as  $a_t$ ;  $E(T) = E(T) \cup \{y_k a_t\}$ ;
10             Rename  $x_i$  as  $x_{i-1}$  for every  $k + 2 \leq i \leq n_1$ ;  $n_1 = n_1 - 1$ ;
11             Continue looking for the next vertex that is not of type 1 in the ordering  $\alpha_0$ , call it  $u$ ;
12         else
13              $E(T) = E(T) \cup \{x_1 y_1, y_1 x_2, x_2 y_2, \dots, x_k y_k, y_k x_{k+1}\}$ ; return  $T$ ;
14     else
15         Let  $u = y_k$  for some  $k$ ;
16         if  $x_i y_{i+1} \in E(G) \forall 1 \leq i \leq (k - 1)$  then
17             Find a vertex which is not of type 2 in the ordering  $\beta_0$  starting from  $y_1$ , call it  $u$ ;
18              $flag = 2$ ;
19         else
20             if  $k \neq n_2$  then
21                  $t = t + 1$ ; rename  $y_k$  as  $a_t$ ;  $E(T) = E(T) \cup \{x_k a_t\}$ ;
22                 Rename  $y_i$  as  $y_{i-1}$  for every  $k + 1 \leq i \leq n_2$ ;  $n_2 = n_2 - 1$ ;
23                 Continue looking for the next vertex that is not of type 1 in the ordering  $\alpha_0$ , call it  $u$ ;
24             else
25                  $E(T) = E(T) \cup \{x_1 y_1, y_1 x_2, x_2 y_2, \dots, y_{k-1} x_k, x_k y_k\}$ ; return  $T$ ;
26     while  $flag == 2$  do
27         if  $u \in Y$  then
28             Let  $u = y_{k+1}$  for some  $k$ ;
29             if  $k + 1 \neq n_2$  then
30                  $t = t + 1$ ; rename  $y_{k+1}$  as  $a_t$ ;  $E(T) = E(T) \cup \{x_k a_t\}$ ;
31                 Rename  $y_i$  as  $y_{i-1}$  for every  $k + 2 \leq i \leq n_2$ ;  $n_2 = n_2 - 1$ ;
32                 Continue looking for the next vertex that is not of type 2 in the ordering  $\beta_0$ , call it  $u$ ;
33             else
34                  $E(T) = E(T) \cup \{y_1 x_1, x_1 y_2, y_2 x_2, \dots, y_k x_k, x_k y_{k+1}\}$ ; return  $T$ ;
35         else
36             Let  $u = x_k$  for some  $k$ ;
37             if  $k \neq n_1$  then
38                  $t = t + 1$ ; rename  $x_k$  as  $a_t$ ;  $E(T) = E(T) \cup \{y_k a_t\}$ ;
39                 Rename  $x_i$  as  $x_{i-1}$  for every  $k + 1 \leq i \leq n_1$ ;  $n_1 = n_1 - 1$ ;
40                 Continue looking for the next vertex that is not of type 2 in the ordering  $\beta_0$ , call it  $u$ ;
41             else
42                  $E(T) = E(T) \cup \{y_1 x_1, x_1 y_2, y_2 x_2, \dots, x_{k-1} y_k, y_k x_k\}$ ; return  $T$ ;

```


Proof. Let T^* be a MIST of G and T be the spanning tree of G returned by Algorithm 2. Recall in the execution of Algorithm 2, we either search for a vertex not of type 1 with the ordering α_0 or we search for a vertex not of type 2 with the ordering β_0 . This is ensured since either we never arrive at line 17 or we arrive at it once and after that flag remains 2 throughout the algorithm. Let a_1, a_2, \dots, a_p be the sequence of vertices encountered in the execution of Algorithm 2. Let X_1 and Y_1 denote the set of vertices visited till a_1 from X and Y side respectively. For $1 < i < p$, let X_i denotes the set of vertices visited from X side after a_{i-1} and upto a_i . Similarly, let Y_i denotes the set of vertices visited from Y side after a_{i-1} and upto a_i . Let X_p and Y_p denote the set of all vertices visited after a_{p-1} from X and Y side respectively.

First suppose Algorithm 2 is searching for a vertex not of type 1 with the ordering α_0 and it never arrives at line 17. This means that flag is 1 throughout the algorithm. To prove that T is a MIST of G , we will prove that T^* can be modified so that it remains a MIST of G and $E(T^*)$ is same as $E(T)$, that is,

$$E(T^*[\bigcup_{j=1}^p X_j \cup \bigcup_{j=1}^p Y_j]) = E(T[\bigcup_{j=1}^p X_j \cup \bigcup_{j=1}^p Y_j]). \quad (2.1)$$

We prove (2.1) using induction on p . If $p = 1$, we have $E(T^*[X_1 \cup Y_1]) = E(T[X_1 \cup Y_1])$ due to Lemma 2.12. Hence, (2.1) is true for $p = 1$. Assume that (2.1) is true for $p = i$.

We now show that (2.1) is true for $p = i + 1$. So, consider vertex a_{i+1} for $i \geq 2$. Two possible cases arise.

Case 1: $a_{i+1} \in X$.

If $a_j \in X$ for each j , $1 \leq j \leq i$, then define $X^* = \bigcup_{j=1}^{i+1} X_j$ and $Y^* = \bigcup_{j=1}^{i+1} Y_j$. Otherwise, let j be the largest index such that $j \in \{1, 2, \dots, i\}$ and $a_j \in Y$. Then define $X^* = \bigcup_{t=j+1}^{i+1} X_t$ and $Y^* = \bigcup_{t=j+1}^{i+1} Y_t$. Note that, in both the cases, we have $N(X^*) = Y^*$.

As $N(X^*) = Y^*$, by Lemma 2.4 we have that the number of pendant vertices from X^* in any spanning tree of G is at least $|X^*| - |Y^*| + 1$. Therefore, $i_{T^*}(X^*) \leq |Y^*| - 1$.

If (2.1) is not true for $p = i + 1$, we remove all edges of T^* who have one end in

$\cup_{j=1}^i (X_j \cup Y_j)$ and the other in $(X_{i+1} \cup Y_{i+1})$ and all edges incident with the vertices of X_{i+1} within T^* . We then add all edges from $E(T[X_{i+1} \cup Y_{i+1}])$ and the edge of T which connects $\cup_{j=1}^i (X_j \cup Y_j)$ to $(X_{i+1} \cup Y_{i+1})$ in T^* . If cycles were created in this process, then we can remove those cycles without introducing more pendant vertices using the method discussed in Lemmas 2.8 and 2.9. Let T_{new}^* denote this updated tree. Define $X' = X \setminus (X^* \cup (\cup_{t=i+2}^p X_t))$ and $Y' = Y \setminus (Y^* \cup (\cup_{t=i+2}^p Y_t))$. We have,

$$\begin{aligned} i(T^*) &= i_{T^*}(X') + i_{T^*}(X^*) + i_{T^*}(\bigcup_{t=i+2}^p X_t) + i_{T^*}(Y') + i_{T^*}(Y^*) + i_{T^*}(\bigcup_{t=i+2}^p Y_t) \\ &\leq i_{T^*}(X') + |Y^*| - 1 + i_{T^*}(\bigcup_{t=i+2}^p X_t) + i_{T^*}(Y') + |Y^*| + i_{T^*}(\bigcup_{t=i+2}^p Y_t) \\ &= i(T_{new}^*), \text{ as } i_{T_{new}^*}(X^*) = |Y^*| - 1. \end{aligned}$$

Thus T_{new}^* is also a MIST of G and (2.1) is true for $p = i + 1$ with $T^* = T_{new}^*$.

Case 2: $a_{i+1} \in Y$. Here, we discuss two subcases.

Subcase 2.1: $a_j \in Y \forall j; 1 \leq j \leq i$.

Here, for $X^* = \cup_{j=1}^{i+1} X_j$ and $Y^* = \cup_{j=1}^{i+1} Y_j$, we have $|X^*| = |Y^*| - i$. As $N(Y^*) = X^*$, by Lemma 2.4 we have that the number of pendant vertices from Y^* in any spanning tree of G is at least $|Y^*| - |X^*| + 1 = i + 1$. Therefore $i_{T^*}(Y^*) \leq |Y^*| - (i + 1)$. Here, $a_1 \in Y$, so, let $a_1 = y_k$ for some k . As we have assumed that flag is 1, this implies that there exists an index t , $1 \leq t \leq k - 1$ such that $x_t y_{t+1} \notin E(G)$. So, for $X' = \{x_1, x_2, \dots, x_t\}$ and $Y' = \{y_1, y_2, \dots, y_t\}$, we have $N(X') = Y'$. Now, by Lemma 2.4, we know that the number of pendant vertices within X' in any spanning tree of G is at least $|X'| - |Y'| + 1 = 1$. So, $i_{T^*}(X') \leq |X'| - 1$, implying that $i_{T^*}(X^*) \leq |X^*| - 1$. If (2.1) is not true for $p = i + 1$, we construct another spanning tree T_{new}^* of G from T^* in the following way: remove all edges of T^* who have one end in $\cup_{j=1}^i (X_j \cup Y_j)$ and the other in $(X_{i+1} \cup Y_{i+1})$ and all edges incident with the vertices of Y_{i+1} within T^* . Then, add all edges from $E(T[X_{i+1} \cup Y_{i+1}])$ and the edge of T which connects $\cup_{j=1}^i (X_j \cup Y_j)$ to $(X_{i+1} \cup Y_{i+1})$ in T^* . As before, if cycles are present, further modify T^* to remove these cycles without introducing more pendant vertices.

Now we have,

$$\begin{aligned} i(T^*) &= i_{T^*}(X^*) + i_{T^*}(X \setminus X^*) + i_{T^*}(Y^*) + i_{T^*}(Y \setminus Y^*) \\ &\leq |X^*| - 1 + i_{T^*}(X \setminus X^*) + |Y^*| - (i + 1) + i_{T^*}(Y \setminus Y^*) = i(T_{new}^*) \end{aligned}$$

Subcase 2.2: $a_j \in X$ for some $1 \leq j \leq i$.

We choose the largest $j \in \{1, 2, \dots, i\}$ such that $a_j \in X$. Then for $X^* = \cup_{t=j+1}^{i+1} X_t$ and $Y^* = \cup_{t=j+1}^{i+1} Y_t$, we have $|X^*| = |Y^*| - (i - j)$. As $N(Y^*) = X^*$, by Lemma 2.4 we have that the number of pendant vertices from Y^* in any spanning tree of G is at least $|Y^*| - |X^*| + 1 = i - j + 1$. Therefore, $i_{T^*}(Y^*) \leq |Y^*| - (i - j + 1)$. If (2.1) is not true for $p = i + 1$, we construct another spanning tree T_{new}^* of G from T^* using the same way as done in subcase 2.1. We have,

$$\begin{aligned} i(T^*) &= i_{T^*}\left(\bigcup_{t=1}^j X_t\right) + i_{T^*}(X^*) + i_{T^*}\left(\bigcup_{t=i+2}^p X_t\right) + i_{T^*}\left(\bigcup_{t=1}^j Y_t\right) + i_{T^*}(Y^*) + i_{T^*}\left(\bigcup_{t=i+2}^p Y_t\right) \\ &\leq i_{T^*}\left(\bigcup_{t=1}^j X_t\right) + |X^*| + i_{T^*}\left(\bigcup_{t=i+2}^p X_t\right) + i_{T^*}\left(\bigcup_{t=1}^j Y_t\right) + |Y^*| - (i - j + 1) + i_{T^*}\left(\bigcup_{t=i+2}^p Y_t\right) \\ &= i(T_{new}^*). \end{aligned}$$

Thus T_{new}^* is also a MIST of G and (2.1) is true for $p = i + 1$ with $T^* = T_{new}^*$.

Hence, we get that (2.1) is true for all p , that is, $E(T^*[X \cup Y]) = E(T[X \cup Y])$ in each possible case, when flag is 1.

If algorithm arrives at line 17, then flag changes to 2 and it remains 2 throughout the algorithm. So, it searches vertex not of type 2 in the ordering β_0 starting from y_1 . There will be analogous arguments for this case also, using Lemmas 2.10 and 2.11 instead. For a quick justification why, with the assumption flag = 1, the above analysis fails if we encounter a vertex, say $u_1 = y_j$, such that u_1 is not type 1 and $x_i y_{i+1} \in E(G)$ for all $1 \leq i \leq (j - 1)$. The analogous failure case for the flag = 2 is, when we encounter a vertex $u_2 = x_k$ that

is not of type 2 and $y_i x_{i+1} \in E(G)$ for all $1 \leq i \leq (k-1)$. Note that these cases cannot simultaneously occur. Otherwise the analysis is symmetric. Consequently, Algorithm 2 returns a maximum internal spanning tree of G . \square

Now, we discuss the running time of Algorithm 2. Suppose Algorithm 2 returns a MIST T . Recall that we visit the vertices in one of the orders $\alpha_0 = (x_1, y_1, x_2, y_2, \dots)$, or $\beta_0 = (y_1, x_1, y_2, x_2, \dots)$. Furthermore, any vertex encountered during the execution of the algorithm must be pendant in T . As we never visit the same vertex twice, these pendant vertices are found in linear-time. The remaining graph must have a Hamiltonian path, and finding the Hamiltonian path is also linear-time in our algorithm. So, all the steps of Algorithm 2 can be executed in $O(n+m)$ time. Hence we have the following corollary.

Corollary 2.14. *A maximum internal spanning tree of a bipartite permutation graph can be computed in linear-time.*

As chain graph is a subclass of bipartite permutation graphs, Algorithm 2 also works for chain graphs. For chain graphs, we computed a bound in terms of the number of edges in its optimal path cover.

2.2.4 Chain Graphs

Let $G = (X, Y, E)$ be a chain graph and (O_X, O_Y) be the chain ordering of G , where $O_X = (x_1, x_2, \dots, x_{n_1})$ and $O_Y = (y_1, y_2, \dots, y_{n_2})$. If a vertex x_i appears before x_j in the chain ordering, we write $x_i < x_j$. In this subsection, we prove the following lower bound for the number of internal vertices in a MIST of G .

Theorem 2.15. *For a chain graph G , let P^* be an optimal path cover of G . Then $\text{Opt}(G) \geq |E(P^*)| - 2$.*

In order to prove Theorem 2.15, we look at optimal path covers of bipartite permutation graphs. Authors of [87] gave an algorithm to find an optimal path cover of a bipartite permutation graph. Note that this algorithm also applies to chain graphs. We recall the algorithm given in [87], but first, we cover some notations used in the algorithm. A path cover $P^* = \{P_1, P_2, \dots, P_k\}$ is *contiguous* if it satisfies the following two conditions:

1. If $x \in X$ is the only vertex in P_i and if $x' < x < x''$, then x' and x'' belong to different paths.
2. If xy is an edge in P_i and $x'y'$ is an edge in P_j , where $i \neq j$ and $x < x'$, then $y < y'$.

A path P is *contiguous* if it is one of the following forms: $x_i y_j x_{i+1} y_{j+1} \dots y_{t-1} x_r$, $x_i y_j x_{i+1} y_{j+1} \dots y_{t-1} x_r y_t$, $y_j x_i y_{j+1} x_{i+1} \dots x_{r-1} y_t x_r$, or $y_j x_i y_{j+1} x_{i+1} \dots x_{r-1} y_t$ such that $r \geq i$ and $t \geq j$. Note that every path in a contiguous path cover is contiguous. Let P be a contiguous path which ends with some edge, say $x_p y_q$. If $y_q x_{p+1} \notin E(G)$, then we say that the path P is not extendable on the right. A contiguous path is said to be a *maximal contiguous path* if it is not extendable on the right. An optimal path cover $P^* = \{P_1, \dots, P_k\}$ is a *maximum optimal path cover* if each P_i covers the maximum number of vertices in $V(G) \setminus \{P_1 \cup P_2 \cup \dots \cup P_{i-1}\}$. According to [87], there exists an optimal path cover which is a maximum optimal path cover for any bipartite permutation graph G such that every path in the path cover is a maximal contiguous path.

As a chain graph is an instance of a bipartite permutation graph, we look at the algorithm from [87] which finds this desired maximum optimal path cover for a chain graph (Algorithm 3). From this point, we refer such a path cover as an optimal path cover of a chain graph.

Algorithm 3: Algorithm for finding an optimal path cover of G

Input: A chain graph $G = (X, Y, E)$ with the ordering of its vertices.

Output: An optimal path cover P of G .

- 1 Mark all vertices in X and Y as not visited; let $P = \emptyset$.
 - 2 **while** all vertices of G are not visited **do**
 - 3 Let x and y be the first vertices in X and Y which are not visited.
 - 4 Let P_x and P_y be the maximal contiguous paths starting from x and y , respectively.
 - 5 $Q := \text{Maximum of } P_x \text{ and } P_y$.
 - 6 $P := P \cup Q$.
 - 7 Mark all vertices in Q as visited.
 - 8 **Output** P .
-

Let P and Q be two distinct paths in a graph G . We define a *combining edge* of P and Q as an edge of G whose one end vertex is in path P and another end vertex is in path Q . Let P^* be the optimal path cover obtained from Algorithm 3. A path in P^* is nontrivial if it has

at least two vertices. We assume that the path components of P^* are ordered with respect to their appearance in Algorithm 3. Before proving Theorem 2.15, we prove some lemmas first.

Lemma 2.16. *Let P and Q be two consecutive nontrivial path components in P^* such that P ends at a vertex of X side and Q starts with a vertex of X side. Then there exists a combining edge of P and Q which joins an internal vertex of P to a pendant vertex of Q .*

Proof. Suppose that P ends at some vertex x and Q starts from some vertex x' , where $x < x'$. Let y be the vertex adjacent to x in P , then $y \in N(x')$ as G is a chain graph. So, the edge yx' is a combining edge for path components P and Q . We see that y is internal in P and x' is pendant in Q . Fig. 2.3 provides an illustration. \square

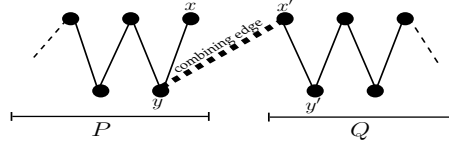


FIGURE 2.3: P ends at X side, Q starts from X side

Lemma 2.17. *There are no consecutive nontrivial path components P and Q in P^* such that:*

1. P ends at a vertex of Y side and Q starts with a vertex of Y side, or,
2. P ends at a vertex of Y side and Q starts with a vertex of X side.

Proof. First, suppose that there are two consecutive nontrivial path components P and Q in P^* such that P ends at a vertex of Y side and Q starts with a vertex of Y side. As P^* was constructed from Algorithm 3, every path component in P^* is maximal contiguous. But, in this case, P is extendable on right. So, this case will not arise. Fig. 2.4 provides an illustration.

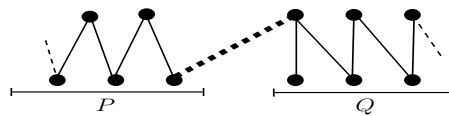


FIGURE 2.4: P ends at Y side, Q starts from Y side

Now, suppose that there are two consecutive nontrivial path components P and Q in P^* such that P ends at a vertex of Y side and Q starts with a vertex of X side. Due to the similar reason, this case will also not arise. Fig. 2.5 provides an illustration. Hence, the

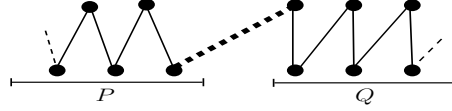


FIGURE 2.5: P ends at Y side, Q starts from X side

lemma holds. □

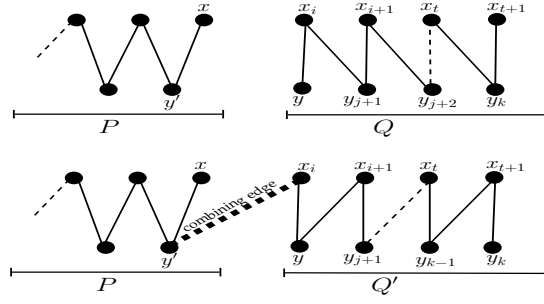
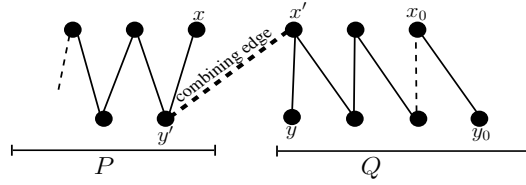
Lemma 2.18. *Let P and Q be two consecutive nontrivial path components in P^* such that P ends at a vertex of X side and Q starts with a vertex of Y side. Then the following is true:*

1. *If Q ends at a vertex of X side, then Q can be modified to another path Q' such that $V(Q) = V(Q')$, Q' is also a maximal contiguous path and there exists a combining edge of P and Q' which joins an internal vertex of P to a pendant vertex of Q' .*
2. *If Q ends at a vertex of Y side, then there exists a combining edge of P and Q which joins an internal vertex of P to an internal vertex of Q .*

Proof. Suppose that P ends at some vertex x and Q starts from some vertex $y = y_j$. Let y' be the neighbor of x in P .

First, suppose that Q ends at a vertex of X side. Let $Q = yx_i y_{j+1} \dots x_t y_k x_{t+1}$. As G is a chain graph, we have that $Q' = x_i y x_{i+1} \dots y_{k-1} x_{t+1} y_k$ is also a path in G . Note that $V(Q) = V(Q')$ and Q' is a maximal contiguous path. We can replace Q with Q' in the path cover P^* . Now we see that edge $y'x_i \in E(G)$ as $N(x) \subseteq N(x_i)$. So, the edge $y'x_i$ is a combining edge for path components P and Q' . We see that y' is internal in P and x_i is pendant in Q' . Fig. 2.6 provides an illustration.

Now, suppose that Q ends at a vertex of Y side. Let x' be the neighbor of y in Q . Since, $x < x'$ and G is a chain graph, edge $y'x' \in E(G)$. Here, we consider the edge $y'x'$ as the combining edge for path components P and Q . We see that y' is internal in P and x' is also internal in Q . Fig. 2.7 provides an illustration.

FIGURE 2.6: P ends at X side, Q starts from Y side and ends at X sideFIGURE 2.7: P ends at X side, Q starts from Y side and ends at Y side

□

Now, we give the proof of the Theorem 2.15.

Proof of Theorem 2.15. Let P^* be the optimal path cover of G obtained from Algorithm 3. Suppose P^* has k path components P_1, P_2, \dots, P_k . Let us denote number of edges of the component P_i by e_i for every $1 \leq i \leq k$. This implies that $e_1 + e_2 + \dots + e_k = |E(P^*)|$. Note that the number of internal vertices in a path with e_i edges is $e_i - 1$.

Let P and Q be two consecutive nontrivial path components in P^* . Then using Lemmas 2.16, 2.17 and 2.18, we see that in each possible case, we get a combining edge of P and Q . If we connect each consecutive nontrivial path component with these combining edges and connect the remaining single vertex components by an arbitrary edge incident with an internal vertex of a nontrivial path component, we obtain a spanning tree of G .

First, assume that we never get P and Q such that P ends at a vertex of X side, Q starts from a vertex of Y side and Q ends at a vertex of Y side. Note then every combining edge connects one internal vertex of P and one pendant vertex of Q . So, $i(T) = e_1 - 1 + e_2 + e_3 + \dots + e_k = |E(P^*)| - 1$.

Now, assume that there exists some P and Q such that P ends at a vertex of X side, Q starts from a vertex of Y side and Q ends at a vertex of Y side. Here, suppose that Q ends at the vertex y_0 and let x_0 be the neighbor of y_0 in Q . We claim that $x_0 = x_{n_1}$. If this is not the case then there exists a vertex x^* in X such that $x^* > x_0$ and $x^* \notin V(Q)$. But, since G is a chain graph, we have that $(y_0, x^*) \in E(G)$ which makes Q , a non-maximal path, a contradiction. Thus, $x_0 = x_{n_1}$ which implies that, if $Q'' \in P^*$ and appears after Q in Algorithm 3, then Q'' is a single vertex path component containing a vertex of Y . This implies that this case appears only once. So, $i(T) = e_1 - 1 + e_2 + e_3 + \dots + e_k - 1 = |E(P^*)| - 2$.

Hence, the number of internal vertices in any MIST of G is at least $|E(P^*)| - 2$, that is, $Opt(G) \geq |E(P^*)| - 2$. \square

Combining Theorem 2.2 and Theorem 2.15, we can state the following corollary.

Corollary 2.19. *For a chain graph G , if P^* denotes an optimal path cover then $Opt(G)$ is either $|E(P^*)| - 1$ or $|E(P^*)| - 2$.*

Now, we give examples of chain graphs which shows that both the bounds (given by Theorem 2.2 and Theorem 2.15) are tight. In Fig. 2.8, G_1 and G_2 are chain graphs and T_1 and T_2 are Maximum Internal Spanning Trees of G_1 and G_2 respectively. We can see that optimal path cover obtained from Algorithm 3 for the graph G_1 is $\{x_1y_1x_2y_2x_3, y_3x_4y_4x_5y_5\}$ which has 8 edges and its MIST T_1 has 6 internal vertices i.e. $Opt(G_1) = |E(P^*)| - 2 = 8 - 2 = 6$. Using Lemma 2.4, it can be verified that any MIST of G_1 has at least four pendant vertices, two from X side and two from Y side; so, G_1 can have at most 6 internal vertices in its MIST. Hence, T_1 is indeed a MIST of G_1 . In a similar manner, optimal path cover obtained from Algorithm 3 for the graph G_2 is $\{x_1y_1x_2y_2x_3, y_3x_4y_4x_5y_5x_6\}$ which has 9 edges and its MIST T_2 has 8 internal vertices i.e. $Opt(G_2) = |E(P^*)| - 1 = 9 - 1 = 8$.

2.3 Relationship between $Opt(G)$ and $|E(P^*)|$

In this section, we summarize the relationship between $Opt(G)$ and $|E(P^*)|$ for all the graph classes discussed in the previous sections.

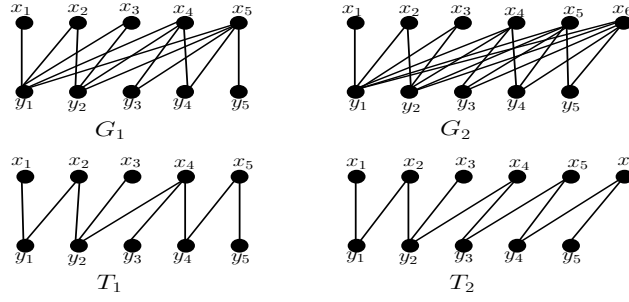


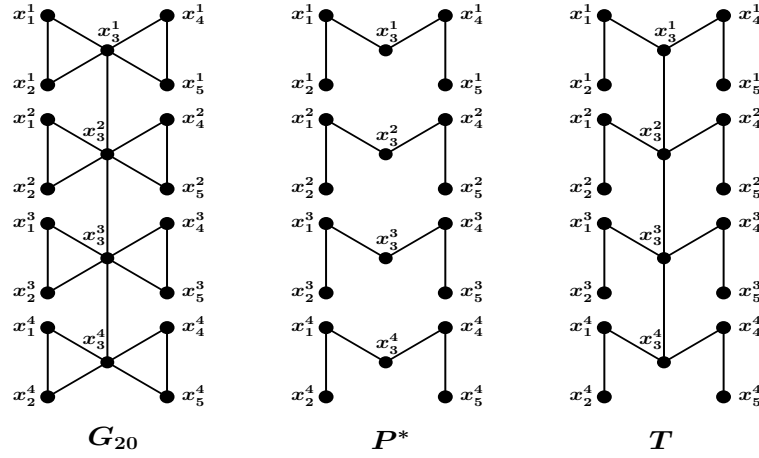
FIGURE 2.8: Examples showing that bounds given by Theorem 2.2 and Theorem 2.15 for chain graphs are tight

2.3.1 Block/Cactus Graph

Let G be a block or cactus graph. Then we show that there does not exist a constant k such that $Opt(G) \geq |E(P^*)| - k$ where P^* is an optimal path cover of G . Recall Corollary 2.6 states that $Opt(G) = n - |Bad(G)|$ and Theorem 2.2 states $Opt(G) \leq |E(P^*)| - 1$. Note that the number of edges in the optimal path cover P^* and the number of components in P^* adds up to n . So, we see that $n - |Bad(G)| = |E(P^*)| - (|Bad(G)| - |P^*|)$. Thus, $Opt(G) = |E(P^*)| - (|Bad(G)| - |P^*|)$ for both block and cactus graphs.

For every integer $n = 5k$ ($k \geq 1$), we construct a connected graph G_n with n vertices and $Opt(G_n) = |E(P^*)| - O(n)$. The graph G_n is both a block graph and a cactus graph as every block of G_n is either an edge or a clique on three vertices. The vertex set of G_n is $V(G_n) = V_1 \cup V_2 \cup \dots \cup V_k$, where $V_i = \{x_1^i, x_2^i, \dots, x_5^i\}$ for each $i \in \{1, 2, \dots, k\}$. The edge set is $E(G_n) = E_1 \cup E_2 \cup \dots \cup E_k \cup E'$, where $E_i = \{x_1^i x_2^i, x_2^i x_3^i, x_3^i x_1^i, x_3^i x_4^i, x_4^i x_5^i, x_5^i x_3^i\}$ for each i and E' contains the edges of the form $x_3^i x_3^{i+1}$ for $1 \leq i \leq (k-1)$. Note $|E(G_n)| = 7k - 1$. We obtain an optimal path cover P^* for G_n having $4k$ edges and k components [74]. The number of bad blocks in G_n is $2k$. Using Theorem 2.5, we obtain a MIST T of G_n with $n - |Bad(G)| = 5k - 2k = 3k$ internal vertices. Thus, $Opt(G_n) = 3k = 4k - k = 4k - \frac{n}{5} = |E(P^*)| - O(n)$. Fig. 2.9 provides an illustration for G_{20} .

Here, we see that $|Bad(G_n)| - |P^*| = 2k - k = k$ which implies that for arbitrary $n = 5k$, we have $Opt(G_n) = |E(P^*)| - k$. So, block and cactus graphs do not have lower bound for $Opt(G)$ of the form $|E(P^*)| - c$ for some fixed natural number c , independent of

FIGURE 2.9: Graph G_{20} , its optimal path cover P^* and its MIST T

n .

2.3.2 Bipartite Permutation Graph

Now, let G be a bipartite permutation graph, then $Opt(G)$ cannot be lower bounded with value $|E(P^*)| - k$ for any fixed natural number k . Below, for every natural number k , we give a construction of a bipartite permutation graph such that $Opt(G) = |E(P^*)| - O(5k)$.

For every integer $n = 5k$ ($k \geq 1$), we construct a connected bipartite permutation graph G_n with n vertices and $Opt(G_n) = |E(P^*)| - O(n)$. For all $1 \leq i \leq k$, let $X_i = \{x_1^i, x_2^i\}$ and $Y_i = \{y_1^i, y_2^i, y_3^i\}$ if i is even and $X_i = \{x_1^i, x_2^i, x_3^i\}$ and $Y_i = \{y_1^i, y_2^i\}$ for odd i . Let $V(G_n) = V_1 \cup V_2 \cup \dots \cup V_k$ where $V_i = X_i \cup Y_i$ for all $1 \leq i \leq k$. Let $E(G_n) = E_1 \cup E_2 \cup \dots \cup E_k \cup E'$ where $E_i = \{xy | x \in X_i, y \in Y_i\}$ for each $1 \leq i \leq k$ and E' is the set of edges of the form $y_2^i x_1^{i+1}$ if i is odd and $x_2^i y_1^{i+1}$ if i is even for each $1 \leq i \leq (k-1)$. We see that G_n is a bipartite permutation graph with n vertices and $n+2k-1$ edges. Algorithm 3 gives an optimal path cover P^* for G_n having $4k$ edges and Algorithm 2 gives a MIST with $3k$ internal vertices. So, we get that $Opt(G_n) = 3k = 4k - k = 4k - \frac{n}{5} = |E(P^*)| - O(n)$. Fig. 2.10 provides an illustration for G_{25} .

Thus $Opt(G)$ for bipartite permutation graphs do not have a lower bound of the form $|E(P^*)| - k$ for some fixed natural number k , independent of n .

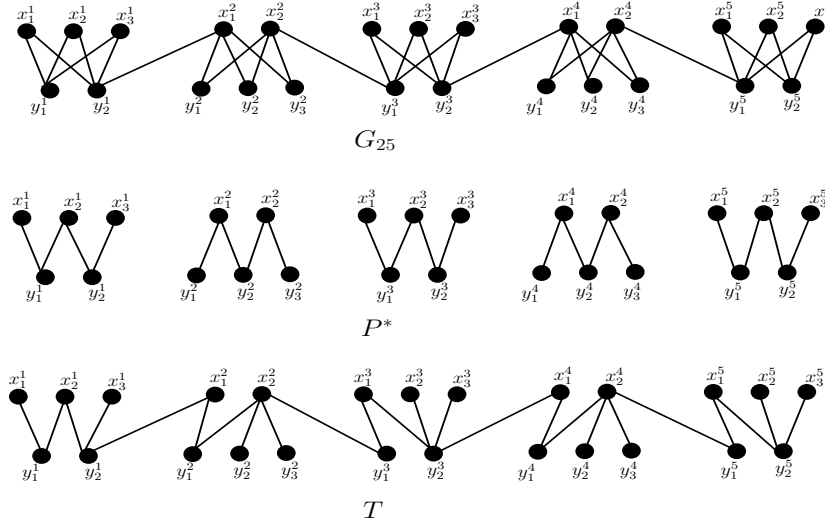


FIGURE 2.10: Graph G_{25} , its optimal path cover P^* from Algorithm 3 and its MIST T from Algorithm 2

2.3.3 Chain Graph and Cographs

In Corollary 2.19, we have proved that $|E(P^*)| - 2 \leq \text{Opt}(G) \leq |E(P^*)| - 1$ for a chain graph G . Figure 2.8 ensures that both these bounds are tight. Thus, we have, $\text{Opt}(G) \in \{|E(P^*)| - 1, |E(P^*)| - 2\}$ when G is a chain graph.

For a cograph G , Theorem 2.7 states that $\text{Opt}(G) = |E(P^*)| - 1$.

2.4 Summary

We studied the MIST problem, a generalization of the HAMILTONIAN PATH problem. As the Decide MIST problem remains NP-hard even for bipartite graphs and chordal graphs [54, 71], we further investigated the complexity of the MIST problem for the following classes of graphs: chain graphs, bipartite permutation graphs, block graphs, cactus graphs and cographs. We found linear-time algorithms for the MIST problem for each of these graph classes.

Li et al. [59] proved an upper bound for $\text{Opt}(G)$ in terms of number of edges in an optimal path cover of G . We further studied this relationship for graph classes mentioned above and proved tight lower bounds for chain graphs and cographs. We also proved that this phenomenon does not hold for general graphs with the construction of a bipartite permutation

graph and a block graph such that $Opt(G)$ is arbitrarily far from $|E(P^*)|$ for the constructed graphs.

Chapter 3

Edge Total Dominating Set

3.1 Introduction

This chapter is dedicated to the study of the *edge total domination* in graphs. We study the Min-ETDS problem from both algorithmic and approximation point of view. We found some subclasses of bipartite and chordal graphs in which the problem is efficiently solvable. From the perspective of approximation, we prove that the problem is APX-hard and provide an approximation algorithm for k -regular graphs, where $k \geq 4$. We also remark on the complexity difference between the Min-EDS problem and the Min-ETDS problem which seem to be closely related.

The notion of *edge total domination* was introduced in 1991 [53]. Kulli et al. [53] discussed the edge total domination number for paths, cycles, complete graphs and complete bipartite graphs. The Decide Min-ETDS problem is NP-complete in bipartite graphs and chordal graphs [75, 92]. We found one subclass of bipartite graphs and two subclasses of chordal graphs for which the problem is in class P. To the best of our knowledge, the only class of graphs for which the Min-ETDS problem can be solved in polynomial-time is the class of trees. In the upcoming sections, we discuss our results in detail. We fix some terminologies first.

For an ETD-set D of a graph $G = (V, E)$, a vertex $u \in V$ is said to be *D-saturated* or *saturated by D*, if u is incident with some edge in D . A set of vertices $A \subseteq V(G)$ is *D-saturated* if every vertex in A is *D-saturated*. The set of all the *D-saturated* vertices of G is denoted by V_D . For a set $A \subseteq V$, an ETD-set D of $G[A]$ is a *saturating set* of A if every vertex of A is incident with some edge in D . The minimum cardinality saturating set of A is called a *minimum saturating set* of A and is denoted by D_A . The following two results will be used in the proofs of theorems in the later sections.

Proposition 8. If G is a graph with $\omega(G) = k$, then for any ETD-set D of G , $|V_D| \geq k - 1$.

Proof. Let D be an ETD-set of G and C be a clique of G such that $|C| = k$. Suppose, to the contrary, that $|V_D| \leq k - 2$. This implies that there exist vertices $u, v \in C$ such that u and v are not saturated by D . Since C is a clique of G , $uv \in E(G)$. In this case, we observe that there is no edge in D which is adjacent to the edge uv . This contradicts the assumption that D is an ETD-set of G . Hence, $|V_D| \geq k - 1$. \square

Proposition 9. Let G be a graph and D be an ETD-set of G . For a vertex u of G , if $u \notin V_D$, then $N_G(u) \subseteq V_D$.

Proof. Let $u \notin V_D$. Suppose, to the contrary, that $N_G(u) \not\subseteq V_D$. This implies that there exists a vertex $v \in N_G(u)$ such that the edge uv is not D -saturated. This contradicts the assumption that D is an ETD-set of G . Hence, $N_G(u) \subseteq V_D$. \square

The section-wise contribution is as follows: In Section 3.2, we present the efficient algorithms to solve the Min-ETDS problem in chain graphs, split graphs, and biconnected proper interval graphs. Next, in Section 3.3 we discuss the results concerning APX-hardness and approximation algorithm. Section 3.4 describes the complexity difference between edge domination and edge total domination.

3.2 Efficient Algorithms

In this section, we present the efficient algorithms to solve the Min-ETDS problem in chain graphs, split graphs, and proper interval graphs with no cut vertices.

3.2.1 Chain Graphs

Let $G = (X, Y, E)$ be a chain graph and $(O_X, O_Y) = (x_1, x_2, \dots, x_{n_1}, y_1, y_2, \dots, y_{n_2})$ be the chain ordering of G . Given a chain ordering of G , for $i < j$, we write $x_i < x_j$ if x_i appears before x_j in the chain ordering. Similarly, for $i < j$, we write $y_i < y_j$, if y_i appears before y_j in the ordering (O_X, O_Y) .

Recall the relation \sim defined on $V(G)$ discussed in Chapter 1. For $u, v \in V(G)$, $u \sim v$ if and only if vertices u and v are open twins. The relation \sim is an equivalence relation.

Let $P_X = \{X_1, X_2, \dots, X_k\}$ be the twin partition of X side and $P_Y = \{Y_1, Y_2, \dots, Y_k\}$ be the twin partition of the Y side. The sets in P_X and P_Y satisfy Propositions 1 and 2.

In particular, $Y_1 = N(X_1)$. As G is a connected chain graph, we see that $y_1x_1 \in E(G)$. Also, $y_1x \in E(G)$ for each $x \in X$ as $N(x_1) \subseteq N(x_i)$ for every $i \geq 2$. In a similar manner, we have that $x_{n_1}y \in E(G)$ for each $y \in Y$. Note that the set of edges $\{y_1x : x \in X\}$ is an ETD-set of G , and so $\gamma'_t(G) \leq n_1$. Similarly, $\{x_{n_1}y : y \in Y\}$ is also an ETD-set of G , and so $\gamma'_t(G) \leq n_2$. Lemma 3.1 combines these observations.

Lemma 3.1. *For a chain graph G , $\gamma'_t(G) \leq \min\{n_1, n_2\}$.*

Through the following set of lemmas, we show the existence of a min-ETD-set D^* of a chain graph $G = (X, Y, E)$, satisfying some specific properties.

Lemma 3.2. *There exists a min-ETD-set D^* of a chain graph G such that the vertices x_{n_1} and y_1 are D^* -saturated.*

Proof. Let G be a chain graph and D be a min-ETD-set of G . Suppose that x_{n_1} is not saturated by D . Since x_{n_1} is not saturated by D , using Proposition 9, we have $Y \subseteq V_D$. Further, as D is a min-ETD-set of G and the set Y is saturated by D , we have $|D| \geq |Y| = n_2$. Also, using Lemma 3.1, we have $|D| \leq n_2$. Therefore, $\gamma'_t(G) = |D| = n_2$ and in this case the set $D^* = \{x_{n_1}y : y \in Y\}$ is a min-ETD-set of G that saturates both vertices x_{n_1} and y_1 .

Suppose next that y_1 is not saturated by D . By Proposition 9, we have $X \subseteq V_D$. Further, as D is a min-ETD-set of G and the set X is saturated by D , we have $|D| \geq |X| = n_1$. By Lemma 3.1, we have $|D| \leq n_1$. Therefore, $\gamma'_t(G) = |D| = n_1$ and in this case the set $D^* = \{y_1x : x \in X\}$ is a min-ETD-set of G that saturates both vertices x_{n_1} and y_1 . Hence, the lemma holds. \square

Lemma 3.3. *There exists a min-ETD-set D of a chain graph G and positive integers i, j such that $i \in [n_1], j \in [n_2]$ satisfying the following:*

$$(a) \quad V_D \cap X = \{x_i, x_{i+1}, \dots, x_{n_1}\}.$$

$$(b) \quad V_D \cap Y = \{y_1, y_2, \dots, y_j\}.$$

(c) $x_i y_j \in E(G)$.

(d) If $i \geq 2$, then $j \geq \max\{t : y_t \in N(x_{i-1})\}$.

Proof. Let G be a chain graph. Using Lemma 3.2, we have a min-ETD-set D^* of G such that x_{n_1} is D^* -saturated. We denote the subgraph $G[D^*]$ of G by G^* . If $X \subseteq V_{D^*}$, then $\gamma'_t(G) = n_1$ and $D = \{y_1 x : x \in X\}$ is a min-ETD-set of G satisfying all the conditions. Similarly, if $Y \subseteq V_{D^*}$, then $\gamma'_t(G) = n_2$ and $D = \{x_{n_1} y : y \in Y\}$ is a min-ETD-set of G satisfying all four conditions (a)-(d). Therefore, we assume that $X \not\subseteq V_{D^*}$ and $Y \not\subseteq V_{D^*}$. Now, if D^* does not satisfy one or more conditions of the lemma, we show that we can modify D^* to get another min-ETD-set of G satisfying all the conditions of the lemma.

(a) Suppose D^* does not satisfy condition (a) of the lemma. Note that $x_{n_1} \in V_{D^*}$. Let r be the largest index such that $x_r \notin V_{D^*}$, implying that, $\{x_{r+1}, x_{r+2}, \dots, x_{n_1}\} \subseteq V_{D^*} \cap X$. Clearly, $r \in [n_1 - 1]$. If $r = 1$, then condition (a) follows with $i = 2$. So, let $r \geq 2$. Now, if $x_j \notin V_{D^*}$ for each j , $j \in [r]$, then we get that $V_{D^*} \cap X = \{x_i, x_{i+1}, \dots, x_{n_1}\}$, where $i = r + 1$ and thus, condition (a) is satisfied. Otherwise, suppose there is a vertex x_j , $j \in [r - 1]$ such that x_j is D^* -saturated. Let $N_{G^*}(x_j) = \{y_{p_1}, y_{p_2}, \dots, y_{p_t}\}$. Since $N(x_j) \subseteq N(x_r)$, we have, $\{y_{p_i} x_r : i \in [t]\} \subseteq E(G)$.

Now, update $D = (D^* \setminus \{x_j y_{p_i} : i \in [t]\}) \cup \{y_{p_i} x_r : i \in [t]\}$. We observe that D is also an ETD-set of G satisfying $|D^*| = |D|$. Thus, D is a min-ETD-set of G such that $\{x_r, x_{r+1}, \dots, x_{n_1}\} \subseteq V_D \cap X$. Moreover, D does not saturate the vertex x_j . By repeatedly applying this modification, we get a min-ETD-set D' of G such that D' does not saturate any vertex x_j , $j < i$ for some $i \in [n_1]$ and, $V_{D'} \cap X = \{x_i, x_{i+1}, \dots, x_{n_1}\}$. Thus, condition (a) is satisfied by D' . Additionally, we observe that $V_{D'} \cap Y = V_{D^*} \cap Y$.

(b) Next, let D be a min-ETD-set of G which satisfies condition (a) and does not satisfy condition (b). By implementing similar modifications to the set D as we did while proving condition (a), we can prove that there exists a min-ETD-set D' of G such that $V_{D'} \cap Y = \{y_1, y_2, \dots, y_j\}$ for some $j \in [n_2]$. Moreover, $V_{D'} \cap X = V_D \cap X$. In addition, D' satisfy both conditions (a) and (b).

(c) From the above discussion, we note that there exists a min-ETD-set of G satisfying the first two conditions of the lemma. Suppose D is a min-ETD-set of G satisfying conditions (a) and (b). If D satisfies condition (c) as well, then we are done. Thus, we may assume that D does not satisfy (c). That is, $x_i y_j \notin E(G)$. From our initial assumptions, $X \not\subseteq V_D$, and so $i \geq 2$. This further implies that $x_{i-1} y_j \notin E(G)$ and $j \geq 2$. Let $y_{j'} \in \{y_1, y_2, \dots, y_j\}$ be the vertex with the maximum index such that $x_{i-1} y_{j'} \in E(G)$. We define the sets $X_0 = \{x_i, x_{i+1}, \dots, x_{n_1}\}$ and $Y_0 = \{y_1, y_2, \dots, y_{j'}\}$. Let $A = X_0 \cup Y_0$ and $B = A \cup Y'$, where $Y' = \{y_{j'+1}, y_{j'+2}, \dots, y_j\}$. Note that $G[A]$ is a complete bipartite subgraph of G . Since D_B is a minimum saturating set of B and D is a min-ETD-set of G such that $V_D = B$, $|D_B| \leq |D|$. Further, we have $|D| = |D_B|$ as D_B is also an ETD-set of G . Now, we modify the set D_B as follows.

First, assume that $|Y_0| \geq 2$. Let $xy' \in D_B$ such that $x \in X_0$ and $y' \in Y'$. If there exists a vertex $y_0 \in Y_0$ such that $xy_0 \notin D_B$, then update the set D_B by $(D_B \setminus \{xy'\}) \cup \{xy_0\}$. Otherwise, if $xy \in D_B$ for each $y \in Y_0$, then update the set D_B by $D_B \setminus \{xy'\}$. Now, suppose that $|Y_0| = 1$ and so, $Y_0 = \{y_1\}$. Again, let $xy' \in D_B$ such that $x \in X_0$ and $y' \in Y'$. Then, either $d_{G[D_B]}(y_1) \geq 2$ or there exists a vertex $x' \in X_0$ such that $x'y_1 \notin D_B$. If $d_{G[D_B]}(y_1) \geq 2$, then update the set D_B by $D_B \setminus \{xy'\}$. But if $x'y_1 \notin D_B$ for some $x' \in X_0$, then update the set D_B by $(D_B \setminus \{xy'\}) \cup \{x'y_1\}$. By implementing these modifications for each edge of $D_B \cap E(G[X_0 \cup Y'])$, we get a new set of edges D' . We claim that D' is a saturating set of A .

First we observe that each vertex of A is saturated by D' . Now, we show that D' is an ETD-set of $G[A]$. For this purpose, we show that, for each $e \in E(G[A])$, there exists an edge in D' which is adjacent to e . So, let e be an edge of the graph $G[A]$. Now, two cases are possible. Suppose firstly that $e \notin D'$. In this case, since end vertices of the edge e are two vertices of A and each vertex of A is saturated by D' , there exists an edge in D' which is adjacent to e . Suppose secondly that $e \in D'$. Assume that $x \in X_0, y \in Y_0$ be the end vertices of e . Further, there can be two cases.

First, suppose that $e \in D_B$. In this case, there exists an edge $e' \in D_B$ which is adjacent to e as D_B is an ETD-set of $G[B]$, where $A \subseteq B$. If e' is incident with the vertex y , then $e' \in D'$. If e' is incident with x and $e' = xy'$, where $y' \in Y_0$, then again we have, $e' \in D'$. But if $e' = xy'$, where $y' \in Y'$, then $e' \notin D'$. In this case, either $xy'' \in D'$ for some

$y'' \in Y_0 \setminus \{y\}$ or $yx' \in D'$ for some $x' \in X_0 \setminus \{x\}$. This is ensured by the modification we performed in D_B . In both ways, we have an edge in D' which is adjacent to e . Thus, if $e \in D_B \cap D'$, then there is an edge in D' which is adjacent to e .

Now, suppose that $e \notin D_B$ but $e \in D'$. Then, since D_B saturates every vertex of Y_0 , there is an edge $e' \in D_B$ incident with y such that $e' \in D'$. Clearly, the edge e' is adjacent to e . Again, if $e \notin D_B, e \in D'$, then there is an edge in D' which is adjacent to e . Therefore, D' is an ETD-set of $G[A]$. Hence, we deduce that $|D_A| \leq |D'| \leq |D_B| \leq |D|$. We also observe that D_A is an ETD-set of G implying further that $|D_A| = |D|$. This yields a min-ETD-set D_A of G which satisfy conditions (a), (b) and (c).

(d) Finally, we have that either $x_i y_j \in E(G)$ or there exists another min-ETD-set of G which satisfies conditions (a), (b) and (c). Condition (d) directly follows from the above discussion and using Proposition 9.

Therefore, the lemma holds. □

Now we present Algorithm 4, which outputs D_A for a given subset of vertices A of G , when $G[A]$ is a complete bipartite graph.

Note that, if D_A is a minimum saturating set of $A \subseteq V(G)$ then each component of $G[D_A]$ is a tree. Algorithm 4 outputs a saturating set D_0 of $A \subseteq V(G)$, and in the subgraph $G[D_0]$, at most one component can be a tree other than a P_3 . So, if, for $A \subseteq V(G)$, D_0 is the saturating set returned by Algorithm 4 such that $G[D_0]$ has c components then either all c components are isomorphic to P_3 , or $c - 1$ components are isomorphic to P_3 and 1 component is a tree that is not a P_3 . Below, we give the proof of correctness of Algorithm 4.

Theorem 3.4. *Given a set $A \subseteq V(G)$ for a chain graph G such that $G[A]$ is a complete bipartite graph, Algorithm 4 outputs a minimum saturating set of A .*

Proof. It is clear that Algorithm 4 outputs a saturating set of A . Let D_1 be the saturating set returned by Algorithm 4 and D_2 be a minimum saturating set of A . We need to show that D_1 is a minimum saturating set of A . On the contrary, suppose that $|D_2| < |D_1|$. Assume that $G[D_1]$ has c components and $G[D_2]$ has t components. Here, $|A| = 3c + (a - 3)$ and

$|D_1| = 2c + (a - 3)$, where $a \geq 3$. So, $|D_2| < 2c + (a - 3)$. Since each component of D_2 is also a tree, we have, $|D_2| = |A| - t = 3c + (a - 3) - t$. This further implies that $3c + (a - 3) - t < 2c + (a - 3)$ and hence, $t > c$. We consider two cases here:

Case 1: All components of $G[D_1]$ are P_3 .

Here, $|A| = 3c$ and $|D_1| = 2c$. So, $|D_2| < 2c$. We also have, $t > c$. As each component of $G[D_2]$ has at least 2 edges, $|D_2| \geq 2t > 2c$, a contradiction on $|D_2| < 2c$. Therefore, $|D_2| = 2c$ and D_1 is a minimum saturating set of A .

Case 2: One component of $G[D_1]$ is not a P_3 .

We denote the component of $G[D_1]$ which is not a P_3 by K . Note that $|V(K)| = a$. Now, we modify D_2 in such a way that at most one component of $G[D_2]$ is a tree other than a P_3 . Let H_1, H_2 be two components of $G[D_2]$ which are not P_3 . Then there exists a subtree T_1 of H_1 which is a P_3 . We remove all edges except the edges of $E(T_1)$ from H_1 . Suppose we removed b number of edges in this process. Then it is possible to connect the remaining vertices of H_1 with H_2 using exactly b new edges as $G[A]$ is a complete bipartite graph and thus, we converted one non P_3 component to a P_3 . After repeated applications of this modification, we get a minimum saturating set of A which satisfy that either all components are P_3 or there is one component that is not a P_3 . We again call this set D_2 . Recall that we have, $t > c$, where t is the number of components of $G[D_2]$ and c is the number of components of $G[D_1]$. Since $t - 1$ components of $G[D_2]$ are P_3 and $c - 1$ components of $G[D_1]$ are P_3 , we have, a subgraph H_1 of $G[D_2]$ such that $|V(H_1)| = |V(K)| = a$ and H_1 has at least two components. Hence, $|D_1| = 2(c - 1) + a - 1$ and $|D_2| = 2(c - 1) + |E(H_1)|$. Thus, $|E(H_1)| < a - 1$. This means that K is a disjoint union of two subgraphs of $G[A]$ such that one of them is a P_3 and Algorithm 4 made the component K which is not a P_3 , a contradiction. Therefore, D_1 is a minimum saturating set of A . \square

Next, we give an algorithm to compute a min-ETD-set of a given chain graph G . Using Lemma 3.3, we know that there exists a min-ETD-set of G such that it saturates each vertex of some special subset A of $V(G)$ and it does not saturate any of the remaining vertices in G . Before formally presenting our Algorithm 5, we present the main ideas in the algorithm for computing a min-ETD-set of G . We first find a minimum saturating set, D_A of all special

Algorithm 4: Algorithm for finding a minimum saturating set of $A \subseteq V(G)$ such that $G[A]$ is a complete bipartite subgraph of G .

Input: A chain graph $G = (X, Y, E)$ and $A \subseteq V(G)$.

Output: D_A : A minimum saturating set of A .

Let $A \cap X = \{a_1, a_2, \dots, a_{p'}\}$ and $A \cap Y = \{b_1, b_2, \dots, b_{q'}\}$;

$D_A = \emptyset, i = 1, j = 1$ and $p = p', q = q'$;

while ($p > 0$ or $q > 0$) **do**

if ($q \geq p > 0$ and $q \geq 2$) **then**

$D_A = D_A \cup \{a_i b_j, a_i b_{j+1}\}$;

$i = i + 1, j = j + 2, p = p - 1, q = q - 2$;

else

if ($q \geq 2$ and $p = 0$) **then**

$D_A = D_A \cup \{a_{i-1} b_j, a_{i-1} b_{j+1}, \dots, a_{i-1} b_{q'}\}$;

$q = 0$;

else

if ($p > q > 0$ and $p \geq 2$) **then**

$D_A = D_A \cup \{a_i b_j, a_{i+1} b_j\}$;

$i = i + 2, j = j + 1, p = p - 2, q = q - 1$;

else

if ($p \geq 2$ and $q = 0$) **then**

$D_A = D_A \cup \{a_i b_{j-1}, a_{i+1} b_{j-1}, \dots, a_{p'} b_{j-1}\}$;

$p = 0$;

else

if $p = 1$ and $q = 1$ **then**

$D_A = D_A \cup \{a_i b_{j-1}, a_{i-1} b_j\}$;

$p, q = 0$;

else

if $p = 1$ and $q = 0$ **then**

$D_A = D_A \cup \{a_i b_{j-1}\}$;

$p = 0$;

else

$D_A = D_A \cup \{a_{i-1} b_j\}$;

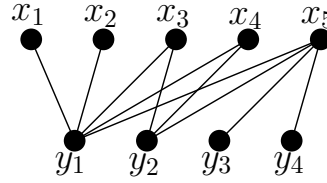
$q = 0$;

Return D_A .

subsets $A \subseteq V(G)$, and then we return the D_A with minimum cardinality. Algorithm 5 correctly outputs a min-ETD-set of G in linear-time. So, we directly state Theorem 3.5.

Theorem 3.5. *Algorithm 5 returns a min-ETD-set of a chain graph in linear-time.*

Now, we describe the Algorithm 5 for a chain graph G shown in Fig. 3.1. For the graph G , Algorithm 5 computes a min-ETD-set in two steps. First, it computes a minimum saturating set of all special subsets A of $V(G)$. Thereafter it returns a D_A with minimum

Algorithm 5: Algorithm for finding a min-ETD-set of a chain graph G **Input:** A chain graph G **Output:** A min-ETD-set of G Compute the partition $P = \{X_1, X_2, \dots, X_k\}$ and $P' = \{Y_1, Y_2, \dots, Y_k\}$ for X and Y side respectively;**for** ($i = n_1$ to 1) **do** Let $A_i = \{x_i, x_{i+1}, \dots, x_{n_1}\}$ and $x_i \in X_s$ for some $s \in [k]$; **if** ($i \geq 2$ and $x_{i-1} \in X_s$) **then** $A_i = A_i \cup \{y_1, y_2, \dots, y_j\}$, where, $j = \max\{r : y_r \in Y_s\}$ Compute D_{A_i} from Algorithm 4; **else if** ($i \geq 2$ and $x_{i-1} \notin X_s$) **then** $A_i = A_i \cup \{y_1, y_2, \dots, y_j\}$, where, $j = \max\{r : y_r \in Y_{s-1}\}$ Compute D_{A_i} from Algorithm 4; **else if** ($i = 1$) **then** $D_{A_i} = \{xy_1 : x \in X\}$ Return a D_{A_i} with minimum cardinality.FIGURE 3.1: A chain graph G .

cardinality. Below, we illustrate the procedure in detail.

In the first step, Algorithm 5 selects $n_1 = 5$ special subsets of $V(G)$ and computes a minimum saturating set for each one of them using Algorithm 4. All 5 subsets ($A_i : i \in [5]$) and corresponding minimum saturating sets are listed below.

i	A_i	D_{A_i}	$ D_{A_i} $
$i = 5$	$A_5 = \{x_5, y_1, y_2\}$	$D_{A_5} = \{y_1x_5, x_5y_2\}$	$ D_{A_5} = 2$
$i = 4$	$A_4 = \{x_4, x_5, y_1, y_2\}$	$D_{A_4} = \{y_1x_4, x_4y_2, y_2x_5\}$	$ D_{A_4} = 3$
$i = 3$	$A_3 = \{x_3, x_4, x_5, y_1\}$	$D_{A_3} = \{y_1x_3, y_1x_4, y_1x_5\}$	$ D_{A_3} = 3$
$i = 2$	$A_2 = \{x_2, x_3, x_4, x_5, y_1\}$	$D_{A_2} = \{y_1x_2, y_1x_3, y_1x_4, y_1x_5\}$	$ D_{A_2} = 4$
$i = 1$	$A_1 = \{x_1, x_2, x_3, x_4, x_5\}$	$D_{A_1} = \{y_1x_1, y_1x_2, y_1x_3, y_1x_4, y_1x_5\}$	$ D_{A_1} = 5$

TABLE 3.1: An illustration of the Algorithm 5.

The second step is to choose a D_A which has minimum number of edges. So, our algorithm returns the set $\{y_1x_5, x_5y_2\}$ as a min-ETD-set of the chain graph G shown in Fig. 3.1.

The Decide Min-ETDS problem remains NP-complete for chordal graphs [92]. In the next two subsections, we study the Min-ETDS problem in two subclasses of chordal graphs, namely, split graphs, and proper interval graphs.

3.2.2 Split Graphs

Here, we discuss the complexity of the Min-ETDS problem in split graphs. It is important to remark that, the decision version of most variations of the vertex domination problem remains NP-complete when restricted to split graphs. In this subsection, we prove that the Min-ETDS problem is linear-time solvable in split graphs. First, we recall the definition of a split graph. A graph $G = (V, E)$ is called a *split graph*, if the vertex set V can be partitioned into two sets K and I , such that K is a clique and I is an independent set of G . Throughout this subsection, $G = (I, K, E)$ denotes a split graph. If there exists a vertex u in K such that $N_G(u) \cap I = \emptyset$, then we update the set I as $I \cup \{u\}$. Thus, we may assume that, for every vertex $u \in K$, $N_G(u) \cap I \neq \emptyset$. Let $|I| = n_1$ and $|K| = n_2$. We observe that a min-ETD-set of a split graph G with $n_2 \leq 3$ can be computed easily. So, we consider split graphs with $n_2 \geq 4$. Before presenting an algorithm, we prove an important lemma.

Lemma 3.6. *If $G = (I, K, E)$ is a split graph, then there exists a min-ETD-set D of G which saturates all the vertices of K .*

Proof. Let D be a min-ETD-set of G such that D does not saturate a vertex $v \in K$. Using Proposition 8, we note that D saturates all the vertices of $K \setminus \{v\}$. Since every vertex of K has a neighbor in I , there exists a vertex $u \in N_G(v) \cap I$. As D is an ETD-set of G , there exists an edge $e = ux \in D$, where $x \in K \setminus \{v\}$. We observe that the set $D' = (D \setminus \{ux\}) \cup \{xv\}$ is an ETD-set of G with $|D'| = |D|$, implying that, D' is also a min-ETD-set of G . Therefore, the result follows. \square

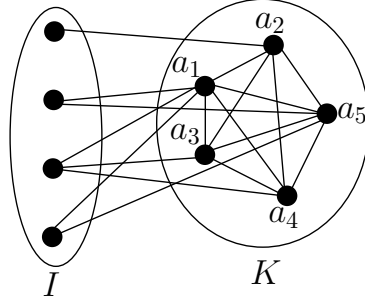
Based on Proposition 8 and Lemma 3.6, we now design Algorithm 6 which computes a min-ETD-set of a given split graph $G = (I, K, E)$. Below, we give the proof of the correctness of Algorithm 6.

Algorithm 6: Algorithm for finding a min-ETD-set of a split graph G **Input:** A split graph $G = (I, K, E)$.**Output:** A min-ETD-set D of G .

```

 $D = \emptyset;$ 
Let  $K = \{a_1, a_2, \dots, a_{n_2}\};$ 
if  $(n_2 \equiv 0 \pmod{3})$  then
   $D = \{a_{3t-2}a_{3t-1}, a_{3t-1}a_{3t} : t \in [\frac{n_2}{3}]\};$ 
else if  $(n_2 \equiv 1 \pmod{3})$  then
   $D = \{a_{3t-2}a_{3t-1}, a_{3t-1}a_{3t} : t \in [\frac{n_2-1}{3}]\} \cup \{a_{n_2-1}a_{n_2}\};$ 
else
   $D = \{a_{3t-2}a_{3t-1}, a_{3t-1}a_{3t} : t \in [\frac{n_2-2}{3}]\} \cup \{a_{n_2-2}a_{n_2-1}, a_{n_2-1}a_{n_2}\};$ 
return  $D.$ 

```

FIGURE 3.2: A split graph $G = (I, K, E)$.

Theorem 3.7. Given a split graph $G = (I, K, E)$, Algorithm 6 computes a min-ETD-set of G in linear-time.

Proof. Let D be the set of edges returned by Algorithm 6. We observe that Algorithm 6 returns a set of edges which induces a collection of paths P_t 's, where $t \geq 3$. Also we note that the set D saturates all vertices of K with a minimum number of edges. Thus, the set D is an ETD-set of G . Due to Lemma 3.6, there is a min-ETD-set of G which saturates all vertices of K . Assume D^* is a min-ETD-set of G such that $K \subseteq V_{D^*}$. Hence, $|D| \leq |D^*|$ implying further that $|D| = |D^*|$. Thus, D is also a min-ETD-set of G . Algorithm 6 selects some edges of $G[K]$ after considering a fix order of vertices of K and outputs those edges in D . This can be done in linear-time. Hence, the theorem holds. \square

Next, we explain the procedure of the Algorithm 6 by a split graph G shown in Fig. 3.2. For this split graph, Algorithm 6 computes a min-ETD-set by selecting some edges from the subgraph $G[K]$ of G in such a manner so that the set of selected edges forms an ETD-set

of G and all vertices of K are saturated by that set. The selection of the edges depends on the remainder we get when $n_2 = |K|$ is divided by 3. Below, we give the computations performed for the graph G shown in Fig. 3.2.

$K = \{a_1, a_2, a_3, a_4, a_5\}$
$n_2 = 5 \equiv 2 \pmod{3}$
$D = \{a_1a_2, a_2a_3, a_3a_4, a_4a_5\}$

TABLE 3.2: An illustration of the Algorithm 6.

Thus, Algorithm 6 returns the set $\{a_1a_2, a_2a_3, a_3a_4, a_4a_5\}$ as a min-ETD-set of G .

3.2.3 Proper Interval Graphs

Let G be a PIG and $\alpha_0 = (v_1, v_2, \dots, v_n)$ be a BCO of G . For $i < j$, $v_i < v_j$ denotes that v_i appears before v_j in α_0 . We use $l(v_i)$ to denote the highest indexed neighbor of v_i in α_0 . Formally if $k = \max\{j : v_j \in N_G[v_i]\}$, then $l(v_i) = v_k$. For a vertex v_i , let $v_k, v_j \in N_G(v_i)$ be two vertices such that $k < i$ and $i < j$, then we call v_k as a *left neighbor* of v_i and v_j as a *right neighbor* of v_i . Further, using the definition of BCO, it can be observed that $v_1v_2 \dots v_n$ is a Hamiltonian path in G . Now, we state an observation which holds true for any PIG.

Observation 1. If $v_iv_j \in E(G)$ and $i < j$, then for every k , $i < k < j$, we have $v_iv_k, v_kv_j \in E(G)$.

For each $i \in [n - 1]$, we observe that there is a clique C of G such that $v_i, v_{i+1} \in C$. Thus, combining Proposition 8 and Observation 1, we note that for $i \in [n]$ at least one of v_i and v_{i+1} will be saturated by every ETD-set of G . Recall that a vertex v of a connected graph G is cut vertex of G if $G - v$ is disconnected. In this subsection, we study the min-ETD-set of proper interval graphs without a cut vertex. First, we have the following important observation.

Observation 2. Let G be a PIG with no cut vertex and $\alpha_0 = (v_1, v_2, \dots, v_n)$ be a BCO of G then $v_{i-1}v_{i+1} \in E(G)$ for every i , $2 \leq i \leq n - 1$.

Throughout this subsection, G denotes a PIG with no cut vertex and its BCO is denoted by $\alpha_0 = (v_1, v_2, \dots, v_n)$. Note that a min-ETD-set of any such PIG when $n \leq 3$, can be computed easily. So, we consider graphs containing at least four vertices. We present the

essential ideas of computing a min-ETD-set of G as follows. We first find the vertex set S of the graph induced by some min-ETD-set of G . Thereafter we find a set of edges D with minimum cardinality saturating all vertices of S . Clearly, D is a min-ETD-set of G . Algorithm 7 computes the vertex set which is saturated by some min-ETD-set of G . Before proving the main results of the subsection, we present the following lemma.

Lemma 3.8. *If D is a min-ETD-set of G and $V_D = \{v_{i_1}, v_{i_2}, \dots, v_{i_k}\}$ such that for $p < q$, $v_{i_p} < v_{i_q}$, where $p, q \in [k]$, then $v_{i_r}v_{i_{r+1}} \in E(G)$ for each $r \in [k-1]$ and $N(u) \subseteq V_D$ for all $u \in V(G) \setminus V_D$.*

Proof. Let $v_{i_r} \in V_D$, where $r \in [k]$. Assume $v_{i_r} = v_j$ for some $j \in [n]$. Now, either $v_{i_{r+1}} = v_{j+1}$ or $v_{i_{r+1}} = v_{j+2}$ because two consecutive vertices will not remain unsaturated by any ETD-set of G . If $v_{i_{r+1}} = v_{j+1}$, then $v_{i_r}v_{i_{r+1}} \in E(G)$ as α_0 is a BCO and G is a proper interval graph. Otherwise, if $v_{i_{r+1}} = v_{j+2}$, we have $v_{i_r}v_{i_{r+1}} \in E(G)$ using Observation 2. Also, using Proposition 9, we have that $N(u) \subseteq V_D$ for all $u \in V(G) \setminus V_D$. Therefore, the lemma follows. \square

Lemma 3.9. *Let $A = \{v_{i_1}, v_{i_2}, \dots, v_{i_k}\} \subseteq V(G)$ satisfying $v_{i_j} < v_{i_r}$ for $j < r$. Let $v_{i_r}v_{i_{r+1}} \in E(G)$ for each $r \in [k-1]$ and $N(u) \subseteq A$ for all $u \in V(G) \setminus A$. If there exists a min-ETD-set D^* of G such that $|V_{D^*}| = k$, then the set D_A is also a min-ETD-set of G .*

Proof. We first show that D_A is also an ETD-set of G . Let xy be an edge of G . If $x \in A$, then there exists an edge $e \in D_A$ which is incident with x as D_A saturates the vertex x . If $e \neq xy$, then e is adjacent to xy in G . Otherwise, if $e = xy$, there exists another edge $e' \in D_A$ which is adjacent to e as D_A is an ETD-set of $G[A]$. Now, let $x \notin A$ which implies that $y \in A$ as $N(u) \subseteq A$ for all $u \in V(G) \setminus A$. Again, since D_A saturates each vertex of A , there exists an edge $e \in D_A$ which is incident with y . Note that $e \neq xy$ as $x \notin A$. So, the edge e is adjacent to the edge xy . Hence, for the edge xy , we always have an edge in D_A which is adjacent to xy . The set D_A is therefore an ETD-set of G .

Next, we show that D_A is a min-ETD-set of G . Let D^* be a min-ETD-set of G such that $|V_{D^*}| = k$. Note that k vertices cannot be saturated by any set of edges containing less than $2\lfloor \frac{k}{3} \rfloor + (k \bmod 3)$ number of edges. This implies that $\gamma'_t(G) = |D^*| \geq 2\lfloor \frac{k}{3} \rfloor + (k \bmod 3)$.

Further, recall that the set D_A saturates all vertices of A ($|A| = k$) using a minimum number of edges and thus, $|D_A| \geq 2\lfloor \frac{k}{3} \rfloor + (k \bmod 3)$. Moreover, a set of edges D_0 , obtained using the similar construction as in Algorithm 8 for the set A , is an ETD-set of $G[A]$. The set D_0 saturates all vertices of A and $|D_0| = 2\lfloor \frac{k}{3} \rfloor + (k \bmod 3)$. Hence, $|D_A| \leq |D_0| = 2\lfloor \frac{k}{3} \rfloor + (k \bmod 3)$. Consequently, $|D_A| = 2\lfloor \frac{k}{3} \rfloor + (k \bmod 3)$, implying that D_A is also a min-ETD-set of G . This proves the result. \square

Now, we give Algorithm 7 that outputs a set of vertices saturated by some min-ETD-set of a proper interval graph with no cut vertex.

Algorithm 7: Algorithm for finding the vertex set which is saturated by some min-ETD-set of G .

Input: A proper interval graph G with no cut vertex.

Output: V_D , for some min-ETD-set D of G .

Initialize: $S = \emptyset, i = 1$;

for ($i = 1$ to n) **do**

$m(v_i) = 0$;

while ($i \leq n$ and $m(v_n) = 0$) **do**

$v_k = l(v_i)$; and $W = \{v_i, v_{i+1}, \dots, v_k\}$;

if (there is some vertex x in W such that $m(x) = 0$) **then**

 Let j be the smallest index such that $v_j \in W$ and $m(v_j) = 0$;

$v_{k'} = l(v_j)$;

$S = S \cup \{v_{j+1}, v_{j+2}, \dots, v_{k'}\}$;

for (each x in S) **do**

$m(x) = 1$;

$m(v_j) = 1$;

$i = i + 1$;

Return S .

Let S be the set of vertices returned by the Algorithm 7. Recall that $\alpha_0 = (v_1, v_2, \dots, v_n)$ denotes a BCO of G . It is clear from the steps of Algorithm 7 that there is no $i \in [n - 1]$ such that $v_i, v_{i+1} \notin S$. We, therefore, state the following result.

Lemma 3.10. *Let $S = \{v_{i_1}, v_{i_2}, \dots, v_{i_k}\}$ be the set of vertices returned by Algorithm 7 such that for $p < q$, $v_{i_p} < v_{i_q}$. If $v_{i_p} = v_j$ for some $j \in [n]$ and $p \in [k - 1]$ then $v_{i_{p+1}} = v_{j+1}$ or $v_{i_{p+1}} = v_{j+2}$.*

Lemma 3.11. *Let $S = \{v_{i_1}, v_{i_2}, \dots, v_{i_k}\}$ be the set of vertices returned by the Algorithm 7 such that for $p < q$ and $v_{i_p} < v_{i_q}$, then $v_{i_r} v_{i_{r+1}} \in E(G)$ for each $r \in [k - 1]$.*

Proof. The proof directly follows from the Observation 2 and Lemma 3.10. \square

Now, we prove the following important lemma which implies the correctness of Algorithm 7.

Lemma 3.12. *There exists a min-ETD-set D^* of G , such that $V_{D^*} = S$, where $S = \{w_1, w_2, \dots, w_k\}$ is the set of vertices returned by Algorithm 7.*

Proof. Let $S^* = \{u_1, u_2, \dots, u_{k'}\}$ be the set of vertices, taken in the order as they appear in α_0 , saturated by some min-ETD-set D^* of G . We also assume that vertices of S are taken in the order as they appear in α_0 . We prove the lemma by modifying the set S^* to get another set S_0 of vertices of G such that $S_0 \subseteq V(G)$ satisfies all conditions of Lemma 3.9. We show that $S_0 = S$ and this further implies that there exists a min-ETD-set D^* of G , such that $V_{D^*} = S$. Now, let $i \in [\min\{k, k'\}]$ be the smallest index such that $u_i \neq w_i$. Let u_i be the vertex v_j in the ordering α . Next, we have two cases to consider.

Case 1: $u_i < w_i$

Here, we have, $v_j \notin S$ and $w_i = v_{j+1}$. Moreover if $i \geq 2$, then $u_{i-1} = w_{i-1} = v_{j-1}$. If $i \neq k'$, then using Lemma 3.10, u_{i+1} is either v_{j+2} or v_{j+1} . Now, if $u_{i+1} = v_{j+2}$, then we modify S^* as follows. We remove the vertex u_i and add the vertex w_i to S^* , that is, $S' = (S^* \setminus \{u_i\}) \cup \{w_i\}$. We do a similar modification when $i = k'$. Now, we show that the set $S' \subseteq V(G)$ satisfies all conditions of Lemma 3.9. Clearly, there is an edge between each pair of consecutive vertices of S' . Now, as $v_j \notin S'$, we need to show that $N_G(v_j) \subseteq S'$.

We note that all left neighbors of v_j were already present in S^* before the modification so they belong to S' also. The vertex v_{j+1} is the only right neighbor of v_j which was absent in S^* prior to the modification. Now, we have $v_{j+1} \in S'$ as $w_i = v_{j+1}$. For each vertex $u \neq v_j$ such that $u \notin S'$, we already have $N(u) \subseteq S'$. Hence, by Lemma 3.9 there exists another min-ETD-set which saturates S' .

Now suppose $u_{i+1} = v_{j+1}$. Among right neighbors of v_j , at most one vertex is not in the set S^* . If $N_G(v_j) \subseteq S^*$, then $S' = S^* \setminus \{u_i\}$. Note that all conditions of Lemma 3.9 are satisfied by the set S' . Suppose next that $v_t \in N_G(v_j)$ is a right neighbor of v_j and $v_t \notin S^*$.

Then, $S' = (S^* \setminus \{u_i\}) \cup \{v_t\}$. Here, the vertex $v_j \notin S'$, so we need to show that all neighbors of v_j must be present in S' . We see that $N_G(v_j) \setminus \{v_t\} \subseteq S^*$, and so $N_G(v_j) \setminus \{v_t\} \subseteq S'$ and due to the modification, we have $v_t \in S'$. For each vertex $u \neq v_j$ such that $u \notin S'$, we have $N(u) \subseteq S'$. Therefore, by Lemma 3.9, there exists another min-ETD-set which saturates S' .

Case 2: $u_i > w_i$

In this case, u_{i-1} is either v_{j-1} or v_{j-2} . Note that $i \neq 1$. If $u_{i-1} = v_{j-1}$, then $w_{i-1} = v_{j-1}$ which further implies that $w_i = v_{j+1}$. This cannot be true as we have considered $u_i > w_i$. Hence, $u_{i-1} = v_{j-2}$, implying $w_{i-1} = v_{j-2}$. Now, the vertex $w_i = v_{j-1}$. We observe, $v_{j-1} \notin S^*$ and $v_{j-2}, v_{j-1} \in S$. This implies that there exists a vertex, say v_t , with $v_t < v_{j-2}$ such that $v_t \notin S$ and $v_tv_{j-2}, v_tv_{j-1} \in E(G)$. Since $u_r = w_r$ for each $r \in [i-1]$, we get $v_t \notin S^*$. This contradicts the fact that D^* is an ETD-set of G because the edge v_tv_{j-1} is not dominated by any edge in D^* . Hence, it is not true that $u_i > w_i$.

Now, we have shown that we can modify S^* to obtain $u_j = w_j$ for every $j \in [i]$. By repeatedly applying the above modifications to S^* , we get a set S_0 such that $u_r = w_r$ for every $r \in [k]$. We can therefore conclude that $S_0 = S$. In addition, the set S satisfies all conditions of Lemma 3.9. Thus, there exists a min-ETD-set D' such that $V_{D'} = S$. Hence, the lemma holds. \square

An Algorithm to compute a min-ETD-set of a PIG G with no cut vertex is given in Algorithm 8.

Algorithm 8: Algorithm for finding a min-ETD-set of G .

Input: A proper interval graph G with no cut vertex.

Output: A min-ETD-set of G .

Let $S = \{w_1, w_2, \dots, w_k\}$ be the set of vertices returned by Algorithm 7;

if $(k \equiv 0 \pmod{3})$ **then**

$D = \{w_{3t-2}w_{3t-1}, w_{3t-1}w_{3t} : t \in [\frac{k}{3}]\}$

else if $(k \equiv 1 \pmod{3})$ **then**

$D = \{w_{3t-2}w_{3t-1}, w_{3t-1}w_{3t} : t \in [\frac{k-1}{3}]\} \cup \{w_{k-1}w_k\};$

else

$D = \{w_{3t-2}w_{3t-1}, w_{3t-1}w_{3t} : t \in [\frac{k-2}{3}]\} \cup \{w_{k-2}w_{k-1}, w_{k-1}w_k\};;$

return D .

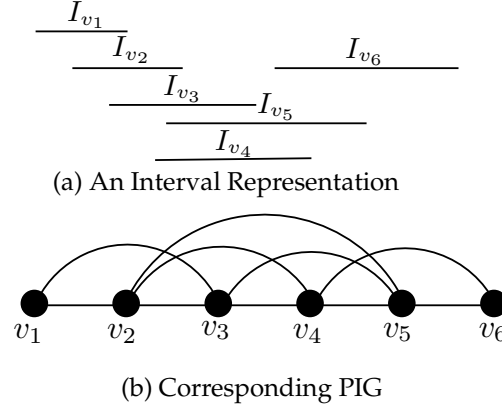


FIGURE 3.3: A PIG with no cut vertex and its interval representation.

We discuss the steps of computing a min-ETD-set of a PIG G shown in Fig. 3.3. Note that G does not have any cut vertices. For this graph, Algorithm 8 computes a min-ETD-set in two steps. First, it computes the vertex set S saturated by some min-ETD-set of G using Algorithm 7. Then it returns a set of edges D so that D is an ETD-set of G and $V_D = S$. Below, we demonstrate the procedure in detail.

For the first step, Algorithm 8 calls Algorithm 7 to find the set S . Algorithm 7 computes S in two iterations. Below, we illustrate those 2 iterations of the algorithm.

Initially: $S = \emptyset$ and $m(v_1) = m(v_2) = m(v_3) = m(v_4) = m(v_5) = m(v_6) = 0$
Iteration 1($i = 1 \leq 6$ and $m(v_6) = 0$) $v_k = v_3, W = \{v_1, v_2, v_3\}$ $v_j = v_1 \in W$ with the smallest index such that $m(v_j) = 0$ $v_{k'} = v_3$ and $S = \{v_2, v_3\}$ $m(v_2) = 1, m(v_3) = 1, m(v_1) = 1$ $i = 2$ and $m(v_6) = 0$
Iteration 2($i = 2 \leq 6$ and $m(v_6) = 0$) $v_k = v_5, W = \{v_2, v_3, v_4, v_5\}$ $v_j = v_4 \in W$ with the smallest index such that $m(v_j) = 0$ $v_{k'} = v_6$ and $S = \{v_2, v_3\} \cup \{v_5, v_6\} = \{v_2, v_3, v_5, v_6\}$ $m(v_5) = 1, m(v_6) = 1, m(v_4) = 1$ $i = 3$ and $m(v_6) = 1$

TABLE 3.3: An illustration of the Algorithm 7.

The second step is to find a set of edges D so that D is an ETD-set of G and $V_D = S$. Algorithm 8 achieves this using the properties satisfied by the edges of $G[S]$ and returns the set $\{v_2v_3, v_3v_5, v_5v_6\}$ as a min-ETD-set of the graph G shown in Fig. 3.3.

The following theorem proves the correctness of Algorithm 8.

Theorem 3.13. *Given a proper interval graph G with no cut vertex, Algorithm 8 computes a min-ETD-set of G in linear-time.*

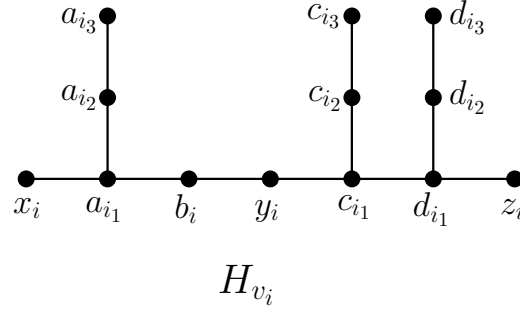
Proof. Algorithm 8 returns an ETD-set of G which saturates each vertex of the set S computed by Algorithm 7. Note that an ETD-set saturating t vertices contains at least $2\lfloor \frac{t}{3} \rfloor + (t \bmod 3)$ edges. Algorithm 8 computes an ETD-set of G which saturates S with exactly $2\lfloor \frac{|S|}{3} \rfloor + (|S| \bmod 3)$ edges. Hence, D , the set returned by Algorithm 8 is a min-ETD-set of G .

To analyze the running time, we see that Algorithm 7 finds the vertex set of a min-ETD-set of G by visiting the vertices in the linear ordering α . Once we have obtained the set S from Algorithm 7, then we select $2\lfloor \frac{|S|}{3} \rfloor + (|S| \bmod 3)$ edges of G in Algorithm 8. All these steps can be implemented in $O(m + n)$ -time. Consequently, the statement holds. \square

3.3 APX-completeness and Approximation Algorithm

In this section, we first prove that the Min-ETDS problem is APX-complete for graphs with maximum degree 3. Further, we propose an approximation algorithm for the problem in k -regular graphs when $k \geq 4$.

For basic definitions and notations on APX-completeness, we refer [90]. To show that an optimization problem $\Pi \in \text{APX}$ is APX-complete, we need to show the existence of an L-reduction from some known APX-complete problem to the problem Π . Note that $\gamma'_t(G) \leq 2\gamma'(G)$ for any graph G . In [26], it is remarked that there is a 2-approximation algorithm to compute an edge dominating set of a given graph. So, a 4-approximation algorithm for the Min-ETDS problem in general graphs is trivial. Hence, the Min-ETDS problem belongs to the class APX. To show the APX-completeness of the Min-ETDS problem, we give an L-reduction from the MINIMUM VERTEX COVER (MVC) problem. The MVC problem asks to find a minimum vertex cover of a given graph G . The MVC problem is already known to be APX-complete for cubic graphs [6].

FIGURE 3.4: A gadget H_{v_i} corresponding to a vertex v_i of V .

Theorem 3.14. *The Min-ETDS problem is APX-complete for graphs with maximum degree 3.*

Proof. We give an L-reduction f from the instances of the MVC problem for cubic graphs to the instances of the Min-ETDS problem for graphs with maximum degree 3. Given a cubic graph $G = (V, E)$, we use the same construction given in [92] to construct an instance $G' = (V', E')$ of the Min-ETDS problem. For sake of completeness, we illustrate the construction here as well.

Let $G = (V, E)$ be a graph, where $V = \{v_1, v_2, \dots, v_n\}$. We replace each vertex v_i of G by a gadget H_{v_i} shown in Fig. 3.4. Let the three edges incident to v_i in G be incident with the three vertices x_i, y_i and z_i in H_{v_i} . According to [91], we note that we can always avoid joining vertices y_i and y_j for two distinct vertices v_i and v_j of G . We observe that the maximum degree of any vertex in G' is 3. We proceed further by proving the following claim.

Claim 3.3.1. If V_c^* is a minimum vertex cover of G , then $|V_c^*| = \gamma'_t(G') - 5n$.

Proof. Let C be a minimum vertex cover of G . Let $D = \{a_{i1}a_{i2}, c_{i1}c_{i2}, d_{i1}d_{i2} : v_i \in V\} \cup \{x_i a_{i1}, y_i c_{i1}, z_i d_{i1} : v_i \in C\} \cup \{a_{i1}b_i, c_{i1}d_{i1} : v_i \notin C\}$. Since C is a vertex cover of G , the set D is an ETD-set of G' . Also, $|D| = 3n + 3|C| + 2(n - |C|) = 5n + |C|$. Hence, $\gamma'_t(G') \leq 5n + |C| = 5n + |V_c^*|$.

Conversely, let D be an ETD-set of G' such that $|D| \leq 5n + |V_c^*|$. We note that $\{a_{i1}a_{i2}, c_{i1}c_{i2}, d_{i1}d_{i2}\} \subseteq D$ for every $v_i \in V$. Let $F_{v_i} = E(H_{v_i}) \setminus \{a_{i1}a_{i2}, c_{i1}c_{i2}, d_{i1}d_{i2}\}$ for each vertex v_i of G . Now, for any vertex v_i of G , the set D contains at least two edges from F_{v_i} to totally dominate the edges $a_{i1}a_{i2}, c_{i1}c_{i2}$ and $d_{i1}d_{i2}$. If it contains exactly two edges

from F_{v_i} , then one of them is $c_{i_1}d_{i_1}$ and the other is $x_ia_{i_1}$ or $a_{i_1}b_i$. If it contains at least three edges from F_{v_i} , then we can replace all of them by the edges $x_ia_{i_1}$, $y_ic_{i_1}$ and $z_id_{i_1}$ in D . Now, let D contains an edge e of the form m_il_j , $i \neq j$, where m_i is x_i, y_i or z_i and l_j is x_j, y_j or z_j .

1. Assume that $m_i = x_i$. If D contains three edges from F_{v_j} , we remove the edge e from D . Otherwise, we replace all edges of $(D \cap F_{v_j}) \cup \{e\}$ by $x_ja_{j_1}$, $y_jc_{j_1}$, $z_jd_{j_1}$.
2. Assume that $m_i = z_i$. Similar modifications can be done in this case also.

Thus, D contains no edge which connects two different gadgets in G' . Clearly, D remains an ETD-set of G' . If D has exactly two edges from some F_{v_i} , then they must be $c_{i_1}d_{i_1}$ and $a_{i_1}b_i$ or $c_{i_1}d_{i_1}$ and $x_ia_{i_1}$ or $c_{i_1}d_{i_1}$ and $a_{i_2}a_{i_3}$. Note that none of the edges $x_ia_{i_1}$, $a_{i_2}a_{i_3}$ dominate the edge b_iy_i and there is no edge between y_i and y_j for any $i \neq j$. So, if D has exactly two edges from some F_{v_i} , then they must be $c_{i_1}d_{i_1}$ and $a_{i_1}b_i$. Therefore, it follows that for every edge e , connecting two different gadgets in G' , D must have six edges from at least one of these gadgets. So, the set $C = \{v_i \in V : D \text{ contains three edges from } F_{v_i}\}$ is a vertex cover of G . Thus, we have, $5n + |V_c^*| \geq |D| \geq \gamma'_t(G') \geq 3|C| + 2(n - |C|) + 3n = 5n + |C|$, implying that $|C| = |V_c^*|$. Hence, $|V_c^*| = \gamma'_t(G') - 5n$. \square

We now return to the proof of Theorem 3.14. It remains to show that f is an L-reduction. Let V_c^* be a minimum vertex cover of G and D_t^* be a min-ETD-set of G' . We have $|D_t^*| = |V_c^*| + 5n$. Since the maximum degree of G is 3 and G is connected, we have $n - 1 \leq m \leq 3|V_c^*|$. This implies that $n \leq 3|V_c^*| + 1$. Consequently, we have $|D_t^*| \leq 21|V_c^*|$.

Now, let D_t be an ETD-set of G' . If D_t contains at least three edges from F_{v_i} for any vertex v_i of G , then we take v_i in the set C . Note that C is a vertex cover of G , and so $|C| \leq |D_t| - 5n$ or, $|D_t| \geq |C| + 5n$. Hence, $|D_t| - |D_t^*| \geq |C| + 5n - |D_t^*| = |C| - (|D_t^*| - 5n) = |C| - |V_c^*|$. Therefore, $|D_t| - |D_t^*| \geq |C| - |V_c^*|$. This proves that f is an L-reduction with $a = 21$ and $b = 1$. \square

Now, we present an approximation algorithm for the Min-ETDS problem in k -regular graphs with $k \geq 4$. For this purpose, first, we note a relationship between the number of

vertices saturated by some min-ETD-set of G and $\gamma'_t(G)$. For designing the algorithm, this relationship is significant. We state this relation formally as follows.

Observation 3. Let S be the vertex set of the graph $G^* = G[D^*]$, where D^* is a min-ETD-set of G , then $|S| \leq \frac{3}{2}|D^*|$.

Proof. Let G^* have k components and let n_i denote the number of vertices in the i^{th} component of G^* , where $i \in [k]$. Since each component of G^* is a tree containing at least two edges of D^* , we have

$$\begin{aligned} |V_{D^*}| &= |V(G^*)| = n_1 + n_2 + \cdots + n_k \\ &= (n_1 - 1) + (n_2 - 1) + \cdots + (n_k - 1) + k \\ &= |D^*| + k \\ &\leq |D^*| + \frac{1}{2}|D^*| \\ &= \frac{3}{2}|D^*| = \frac{3}{2}\gamma'_t(G). \end{aligned}$$

□

Now, we design an approximation algorithm for the Min-ETDS problem in k -regular graphs with $k \geq 4$ using a well studied problem, the MINIMUM CONNECTED VERTEX COVER (MCVC) problem. This problem is a variation of the well known MVC problem. A vertex cover C of a graph G such that $G[C]$ is connected is a *connected vertex cover* of G . Given a graph G , the MCVC problem is to find a connected vertex cover of minimum cardinality. A *minimum connected vertex cover* (min-CVC) of G is connected vertex cover of minimum cardinality. We proceed further by proving an important relationship between a min-ETD-set and a min-CVC of a graph G , which will enable us to design our algorithm.

Theorem 3.15. *If D^* is a min-ETD-set of G and S^* be a min-CVC of G , then $|S^*| \leq 2|D^*|$, that is, $|S^*| \leq 2\gamma'_t(G)$.*

Proof. Let $S = V_{D^*}$. Note that S is a vertex cover of G . Let G' denote the subgraph of G induced by the vertex set S , that is, $G' = G[S]$. We see that $V(G) \setminus V(G')$ is an independent set of G . Hence, no two vertices of $V(G) \setminus V(G')$ are adjacent.

If G' is connected, then $V(G') = S$ is a connected vertex cover of G . Hence we may assume that G' is disconnected. Let G' have $k \geq 2$ components, namely G_1, G_2, \dots, G_k . Since G is connected, we have that, for some $i \neq j$ and for $x \in V(G_i), y \in V(G_j)$, there is a path P between x and y in G . Let $P: u_1 u_2 \dots u_t$ where $u_1 = x$ and $u_t = y$. As $i \neq j$, all vertices from the set $\{u_2, u_3, \dots, u_{t-1}\}$ do not belong to S . So, there exists a vertex $u_r \in \{u_2, u_3, \dots, u_{t-1}\}$ such that $u_r \in V(G) \setminus S$ and $u_{r-1}, u_{r+1} \in S$. Now, we consider the set $S' = S \cup \{u_r\}$ and we update the graph G' by considering $G' = G[S']$. Hence, the number of components in the resulting graph G' is now $k - 1$. We proceed in this manner with similar modifications until the resulting graph G' is connected.

Hence, we may assume that G' is connected. Letting $S' = V(G')$ yields $|S'| \leq |S| + k - 1 < |S| + k \leq |S| + \frac{1}{2}|D^*| \leq \frac{3}{2}|D^*| + \frac{1}{2}|D^*| = 2|D^*|$. The last inequality holds due to Observation 3. Note that S' is a connected vertex cover of G and thus, $|S^*| \leq 2|D^*|$, that is, $|S^*| \leq 2\gamma'_t(G)$. \square

We finally present our promised approximation algorithm.

Algorithm 9: Algorithm for finding a min-ETD-set of a k -regular graph G

Input: A k -regular graph $G, k \geq 4$.

Output: A min-ETD-set of G . Compute a connected vertex cover S of G using the algorithm in [62];

Let $G' = G[S]$ and $D' = E(G')$;

while (G' contains at least one cycle) **do**

Let C be a cycle in G' ;
 Let $e \in E(C)$;
 $D' = D' \setminus \{e\}$;
 $G' = G[D']$;

return D' .

The approximation ratio and time complexity of Algorithm 9 is proved in the following theorem.

Theorem 3.16. *Algorithm 9 is a $\frac{4k}{k+2} + O(\frac{1}{n})$ -approximation algorithm for Min-ETDS problem in k -regular graphs for $k \geq 4$ and it runs in $O(n^3)$ -time.*

Proof. Assume that S^* is a min-CVC of G and D^* is a min-ETD-set of G . Let S be the connected vertex cover of G obtained by the algorithm in [62] and let D' be the edge set returned by Algorithm 9. It is clear from Algorithm 9 that D' is an ETD-set of G as it is obtained by removing one edge from each cycle of a connected graph. Since D' has no cycle and $G[D']$ is connected, it is a tree. Hence,

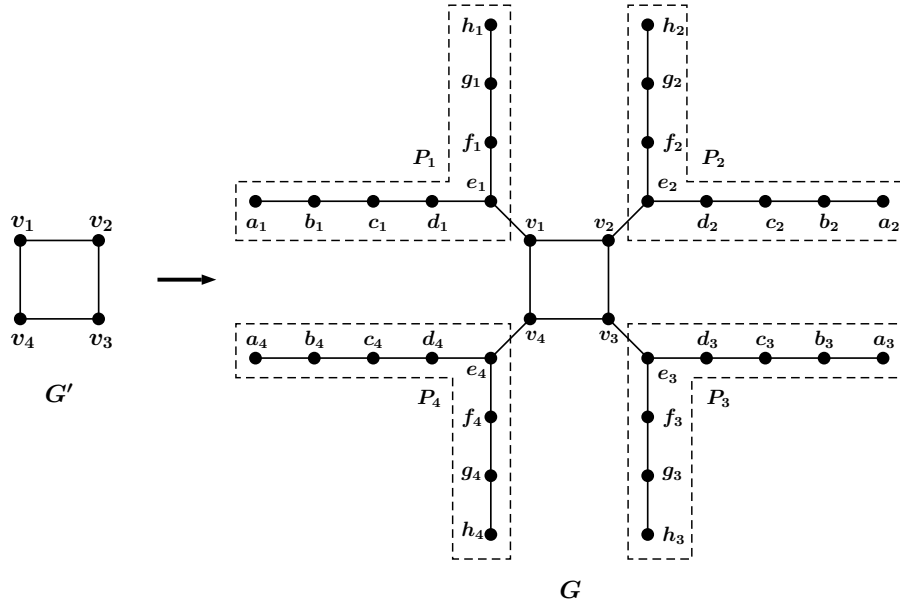
$$\begin{aligned} |D'| = |S| - 1 &\leq \left(\frac{2k}{k+2} + O\left(\frac{1}{n}\right)\right)|S^*| - 1 \\ &\leq \left(\frac{2k}{k+2} + O\left(\frac{1}{n}\right)\right)2|D^*| \\ &= \left(\frac{4k}{k+2} + O\left(\frac{1}{n}\right)\right)|D^*|. \end{aligned}$$

The second inequality holds using Theorem 3.15. Hence, Algorithm 9 is a $\frac{4k}{k+2} + O(\frac{1}{n})$ -approximation algorithm.

To analyze the time complexity, we see that Algorithm 9 first finds a CVC S of G using the algorithm in [62]. The authors of [62] have proved that their algorithm runs in $O(n^3)$ -time. After that, Algorithm 9 looks at all cycles in the graph $G[S]$ and removes one edge from each of them so that the updated graph has no cycles and is connected. Finally, it returns the set of edges of the updated graph. It is possible to accomplish this work in linear-time. Thus, Algorithm 9 runs in $O(n^3)$ -time. \square

3.4 Complexity Difference Between Edge Domination and Edge Total Domination

In this section, we discuss the complexity difference between the Min-EDS and the Min-ETDS problem. Although these two problems appear to be firmly related, they differ in terms of complexity. For this purpose, first, we define a new graph class, namely GP_8 graphs. We

FIGURE 3.5: A GP_8 graph obtained from C_4 .

show that for the class of GP_8 graphs, the Decide Min-ETDS problem is NP-complete, while the edge domination number can be computed efficiently for any GP_8 graph.

Before presenting the definition of a GP_8 graph, we explain some terminologies that are going to be used in this section. For a graph $G = (V, E)$ and an edge $e \in E$, we say that the edge e *dominates* an edge e_0 of G if $e_0 \in \{f : f \text{ is adjacent to } e\} \cup \{e\}$. An edge e *totally dominates* another edge f if the edges e and f are adjacent edges.

Definition 3.17. A graph $G = (V, E)$ is called a GP_8 graph if it is obtained from a general graph $G' = (V', E')$, where $V' = \{v_1, v_2, \dots, v_n\}$, by adding a path $P_i : a_i b_i c_i d_i e_i f_i g_i h_i$ on 8 vertices to each vertex $v_i \in V'$ and the edge $v_i e_i$ for all $i \in [n]$. Formally, $V = V' \cup \{a_i, b_i, c_i, d_i, e_i, f_i, g_i, h_i : i \in [n]\}$ and $E = E' \cup \{v_i e_i, a_i b_i, b_i c_i, c_i d_i, d_i e_i, e_i f_i, f_i g_i, g_i h_i : v_i \in V \text{ and } i \in [n]\}$.

An example of a GP_8 graph obtained from C_4 , a cycle on four vertices, is shown in Fig. 3.5. Now we prove the following results for GP_8 graphs.

Theorem 3.18. If G is a GP_8 graph obtained from a general graph $G' = (V', E')$ with $|V'| = n$, then $\gamma'(G) = 3n$.

Proof. Let $G = (V, E)$ be a GP_8 graph obtained from a general graph $G' = (V', E')$.

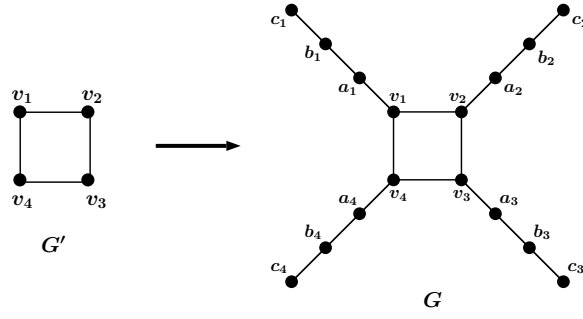
From Definition 3.17, we note that if $V' = \{v_1, v_2, \dots, v_n\}$, then we have $V = V' \cup \{a_i, b_i, c_i, d_i, e_i, f_i, g_i, h_i : i \in [n]\}$ and $E = E' \cup \{v_i e_i, a_i b_i, b_i c_i, c_i d_i, d_i e_i, e_i f_i, f_i g_i, g_i h_i : v_i \in V \text{ and } i \in [n]\}$. We note that the set $\{b_i c_i, f_i g_i, v_i e_i : i \in [n]\}$ is an edge dominating set of G . Therefore, if D is a minimum edge dominating set of G , then $|D| \leq 3n$.

Further, suppose D is a minimum edge dominating set of G . We note that, to dominate the edge $a_i b_i$, either $a_i b_i \in D$ or $b_i c_i \in D$. Without loss of generality we assume that $b_i c_i \in D$ and $a_i b_i \notin D$ for each $i \in [n]$. Similarly, to dominate the edge $g_i h_i$, either $g_i h_i \in D$ or $f_i g_i \in D$. Again, without loss of generality we assume that $f_i g_i \in D$ and $g_i h_i \notin D$ for each $i \in [n]$. Now, it may be noted that the edge $d_i e_i$ is neither dominated by $b_i c_i$ nor by $f_i g_i$. Also, the edge $d_i e_i$ can only be dominated by an edge in the set $S_i = \{d_i e_i, e_i f_i, v_i e_i, c_i d_i\}$, where $i \in [n]$. Thus, if $d_i e_i$ is dominated by $e \in D \cap S_i$ then the set $D = (D \setminus \{e\}) \cup \{v_i e_i\}$ is also a minimum edge dominating set of G . Hence, there exists a min-ETD-set D such that $b_i c_i, f_i g_i, v_i e_i \in D$ for each $i \in [n]$. Therefore, $|D| \geq 3n$. Consequently, $\gamma'(G) = 3n$. \square

Lemma 3.19. *If G is a GP_8 graph obtained from the graph G' , then G' has an ETD-set of size at most k if and only if G has an ETD-set of size at most $k + 4n$.*

Proof. Let $G = (V, E)$ be a GP_8 graph obtained from the graph $G' = (V', E')$. If D' is an ETD-set of G' of size at most k , then the set $D = D' \cup \{b_i c_i, c_i d_i, e_i f_i, f_i g_i : i \in [n]\}$ is an ETD-set of G of size at most $k + 4n$.

Conversely, suppose D is an ETD-set of the graph G of size at most $k + 4n$. We note that, to totally dominate the edge $a_i b_i$, the edge $b_i c_i \in D$ for each $i \in [n]$. To totally dominate the edge $b_i c_i$, at least one of $a_i b_i$ and $c_i d_i$ belongs to D , implying that $|D \cap \{a_i b_i, b_i c_i, c_i d_i\}| \geq 2$ for each $i \in [n]$. Similarly, to totally dominate the edge $g_i h_i$, the edge $f_i g_i \in D$, and to totally dominate the edge $f_i g_i$, at least one of $e_i f_i$ and $g_i h_i$ belongs to D , implying that $|D \cap \{e_i f_i, f_i g_i, g_i h_i\}| \geq 2$ for each $i \in [n]$. Therefore, adopting the notation from Definition 3.17, we have $|D \cap E(P_i)| \geq 4$ for each $i \in [n]$. Further no edge of E' can be dominated by the edges in $E(P_i)$ for any $i \in [n]$. Let $D' = D \setminus (\bigcup_{i=1}^n E(P_i))$. Now suppose $v_i e_i \in D'$. If all edges incident with v_i different from $v_i e_i$ are in D' , then remove $v_i e_i$ from D' ;

FIGURE 3.6: A GP_3 graph obtained from C_4 .

otherwise in D' , we replace $v_i e_i$ with some other edge e incident on v_i such that $e \in E' \setminus D'$. The resulting set D' is an ETD-set of G' satisfying $|D'| \leq k$. Hence, the result follows. \square

Theorem 3.20. *The Decide Min-ETDS problem is NP-complete for GP_8 graphs.*

Proof. The proof directly follows from Lemma 3.19. \square

Now we define another graph class, namely GP_3 graphs. For this graph class, we show that the decision version of the Min-EDS problem is NP-complete but the Min-ETDS problem is efficiently solvable.

Definition 3.21. A graph $G = (V, E)$ is called a GP_3 graph if it is obtained from a general graph $G' = (V', E')$, where $V' = \{v_1, v_2, \dots, v_n\}$, by adding a path $P_i = a_i b_i c_i$ on three vertices corresponding to each vertex $v_i \in V'$ and adding an edge $v_i a_i$ for all $i \in [n]$. Formally, $V = V' \cup \{a_i, b_i, c_i : i \in [n]\}$ and $E = E' \cup \{v_i a_i, a_i b_i, b_i c_i : v_i \in V' \text{ and } i \in [n]\}$.

An example of a GP_3 graph obtained from a cycle on four vertices is shown in Fig. 3.6.

Theorem 3.22. *If G is a GP_3 graph obtained from the graph $G' = (V', E')$, where $|V'| = n$, then $\gamma'_t(G) = 2n$.*

Proof. Let $G = (V, E)$ be a GP_3 graph obtained from a general graph $G' = (V', E')$. From Definition 3.21, we note that if $V' = \{v_1, \dots, v_n\}$ then we have $V = V' \cup \{a_i, b_i, c_i : i \in [n]\}$ and $E = E' \cup \{v_i a_i, a_i b_i, b_i c_i : v_i \in V' \text{ and } i \in [n]\}$. The set $D' = \{v_i a_i, a_i b_i : i \in [n]\}$ is an ETD-set of G . Therefore, we have $\gamma'_t(G) \leq |D'| = 2n$.

Next, assume that D is a min-ETD-set of G . To totally dominate the edge $b_i c_i$, the edge $a_i b_i \in D$. Moreover to totally dominate the edge $a_i b_i$, the set D contains at least one of the edges $b_i c_i$ and $a_i v_i$, implying that $|D \cap \{v_i a_i, a_i b_i, b_i c_i : i \in [n]\}| \geq 2$. Therefore, $|D| \geq 2n$. Consequently, we have $\gamma'_t(G) = 2$. \square

Lemma 3.23. *If G is a GP_3 graph obtained from the graph G' , then G' has an edge dominating set of size at most k if and only if G has an edge dominating set of size at most $k + n$.*

Proof. Let $G = (V, E)$ be a GP_3 graph obtained from a general graph $G' = (V', E')$. Every ED-set of G' can be extended to an ED-set of G by adding to it the edges from the set $\{a_i b_i : i \in [n]\}$. Hence, if G' has an ED-set of size at most k , then G has an ED-set of size at most $k + n$.

Conversely, assume that D is an ED-set of G of size at most $k + n$. To dominate the edge $b_i c_i$, we note that $b_i c_i \in D$ or $a_i b_i \in D$. Thus, $|D \cap \{a_i b_i, b_i c_i\}| \geq 1$ for each $i \in [n]$. Further, no edge of E' can be dominated using an edge in the set $\{a_i b_i, b_i c_i : i \in [n]\}$. Let $D' = D \setminus \{a_i b_i, b_i c_i : i \in [n]\}$. Now, if $D' \subseteq E'$, then D' is an ED-set of G' and $|D'| \leq k$. Otherwise, assume that $v_i a_i \in D'$ for some $i \in [n]$. If all edges incident with v_i other than $v_i a_i$ are in D' , then remove $v_i a_i$ from D' ; otherwise, replace the edge $v_i a_i$ in D' with some other edge $e \in E' \setminus D'$ incident with v_i . In both cases, the set D' is an ED-set of G' and $|D'| \leq k$. Hence, the result follows. \square

Theorem 3.24. *The decision version of the Min-EDS problem is NP-complete for GP_3 graphs.*

Proof. The proof directly follows from Lemma 3.23. \square

3.5 Summary

We studied the computational complexity of the Min-ETDS problem. We discussed the complexity difference between the Min-EDS and the Min-ETDS problem. We resolved the complexity status of the problem in chain graphs, a subclass of bipartite graphs. Further,

we studied the problem in two subclasses of chordal graphs, split graphs, and proper interval graphs without a cut vertex.

We also studied the approximation hardness and approximation algorithm for the problem. We proved that the problem is APX-complete for graphs with maximum degree 3 and designed an efficient approximation algorithm for k -regular graphs when $k \geq 4$. We remark that the key to design our approximation algorithm is a relationship between cardinalities of a min-ETD-set and a min-CVC of a graph.

Chapter 4

Grundy (Double) Dominating Sequence

4.1 Introduction

This chapter concentrates on two distinct types of vertex sequences in graphs, namely “dominating sequences” and “double dominating sequences”. For a given graph G , the aim is to find a dominating sequence (double dominating sequence) of maximum length. The chapter presents algorithmic and hardness results for both the optimization problems.

Domination in graphs is one of the classical problems in graph theory and has an extremely rich literature. The MINIMUM DOMINATION problem is to find a dominating set of minimum cardinality. This problem and its variations have numerous applications in real-world problems including social networks, facility location problems, and routing problems. For more information about domination in graphs, the reader is referred to three recent monographs [13, 14, 35]. To find a dominating set of minimum cardinality in a given graph, it might seem reasonable to pick vertices one at a time depending on some criterion, such as large vertex degree.

The domination game, introduced in 2010 ([23]) models this vertex-by-vertex approach of building a dominating set of a graph. In this game, we have a graph G and two players, Dominator and Staller, who alternately take turns and choose a vertex from G such that whenever a vertex is chosen by either player, at least one additional vertex of G is dominated that was not dominated by the previously chosen vertices. The game begins with either Dominator or Staller taking the first move, and they continue to alternate turns. The game comes to an end when there is no more vertex to choose. Dominator and Staller have the opposite goals in the game: one wants the game to end in as few moves as possible while the other one wants to extend the length of the game. At the end of the game, the set of vertices chosen during the game forms a dominating set. Therefore, throughout the game, a sequence of vertices is selected. In several papers, authors explored a sequence obtained by the same

basic rule, yet assuming that only the slow player plays the game, leading to the following specific definitions.

Let $S = (v_1, v_2, \dots, v_k)$ be a sequence of distinct vertices of G . The corresponding set $\{v_1, v_2, \dots, v_k\}$ of vertices from the sequence S will be denoted by \widehat{S} . A sequence $S = (v_1, v_2, \dots, v_k)$ is a *closed neighborhood sequence* if $N[v_i] \setminus \bigcup_{j=1}^{i-1} N[v_j] \neq \emptyset$, holds for every $i \in \{2, 3, \dots, k\}$. If, in addition, \widehat{S} is a dominating set of G , then S is a *dominating sequence* of G . Clearly, the length k of a dominating sequence S is bounded below by the domination number, $\gamma(G)$, of G . A dominating sequence of maximum length in G is called a *Grundy dominating sequence* (GD-sequence) of G . The cardinality of such a sequence is called the *Grundy domination number* of G and is denoted by $\gamma_{gr}(G)$. Given a graph G , the GRUNDY DOMINATION (GD) problem asks to find a Grundy dominating sequence of G . By the GDD problem, we mean the decision version of the GD problem.

These concepts were introduced and studied in 2014 by Brešar, Golobranec, Milanič, Rall, and Rizzi [20], where the motivation came from the domination game as described above. In addition, Grundy domination presents the worst-case scenario in the process of the online update of a dominating set in the expanding network. In 2021, domination games, as well as Grundy domination, were comprehensively surveyed in the book [21].

A close relation between dominating sequences in graphs and covering sequences in hypergraphs was found in the seminal paper [20]. A hypergraph is a generalization of a graph. Mathematically, a hypergraph is represented by an ordered pair $H = (X, \mathcal{E})$, where X is the set of vertices of H and \mathcal{E} is the set of hyperedges of H . A hyperedge of H is a subset of vertices of H . Given a hypergraph $H = (X, \mathcal{E})$ with no isolated vertices, an *edge cover* of H is a set of hyperedges from \mathcal{E} that covers all vertices of X . That is, the union of the hyperedges from an edge cover is the vertex set X . An edge cover of H having minimum number of hyperedges is called a *minimum edge cover* of H . The cardinality of such an edge cover of H is the *edge covering number* of H and is denoted by $\rho(H)$.

Now, consider a sequence of hyperedges $\mathcal{C} = (C_1, \dots, C_r)$ of a hypergraph H . If for each i , $i \in [r]$, C_i covers a vertex not covered by C_j , for all $j < i$, then, \mathcal{C} is a *legal hyperedge sequence* of H . If $\mathcal{C} = (C_1, C_2, \dots, C_r)$ is a legal hyperedge sequence and the

set $\widehat{\mathcal{C}} = \{C_1, C_2, \dots, C_r\}$ is an edge cover of H , then \mathcal{C} is an *edge covering sequence*. An edge covering sequence of maximum length in H is a *Grundy covering sequence* of H . The size of such a sequence is the *Grundy covering number* of H and is denoted by $\rho_{gr}(H)$. Given a hypergraph $H = (X, \mathcal{E})$, the GRUNDY COVERING problem asks to find an edge covering sequence of H having size $\rho_{gr}(H)$. The GRUNDY COVERING DECISION (GCD) problem is the decision version of the GRUNDY COVERING problem. It was shown in [20] that the GDD problem is NP-complete by reduction from the GCD problem, while for the NP-completeness of the GCD problem, a reduction from the classical FEEDBACK ARC SET problem was used.

Recently Haynes et al. proposed various kinds of vertex sequences, each of which is specified in terms of some conditions that must be satisfied by every subsequent vertex in the sequence [44]. Predictably, double domination in the sequence context is one of these variations. Recall that, for a graph G with no isolated vertices, a set $D \subseteq V$ is called a *double dominating set* of G , if for every vertex $x \in V$, $|N_G[x] \cap D| \geq 2$.

A sequence S is called a *double neighborhood sequence* of a graph $G = (V, E)$ without any isolated vertices, if for each $i \geq 2$, the vertex v_i dominates at least one vertex u in V which is dominated at most once by the vertices v_1, v_2, \dots, v_{i-1} . If \widehat{S} is a double dominating set of G , then we call S a *double dominating sequence* of G . A double dominating sequence of G with maximum length is called a *Grundy double dominating sequence* (GDD-sequence) of G . The length of a GDD-sequence is the *Grundy double domination number* of G and is denoted by $\gamma_{gr}^{\times 2}(G)$. Given a graph G with no isolated vertices, the GRUNDY DOUBLE DOMINATION (GD2) problem asks to find a GDD-sequence sequence of G . By the GD2D problem, we mean the decision version of the GD2 problem. This concept was introduced in a slightly different manner by Haynes et al. in [44]. In their version, S_i denotes the subsequence (v_1, v_2, \dots, v_i) which consists of the first i vertices of the sequence $S = (v_1, v_2, \dots, v_k)$. We state their definition below.

Definition 1. Let $S = (v_1, v_2, \dots, v_k)$ be a sequence of vertices in a graph G without isolated vertices such that every vertex $v_i, i \geq 2$ dominates at least one vertex x in $V \setminus \widehat{S_{i-1}}$ that is dominated at most once by vertices in $\widehat{S_{i-1}}$. Such a sequence is called a *double neighborhood sequence* of G . A double neighborhood sequence of maximal length is called a *double*

dominating sequence of G . The maximum length of a double dominating sequence is called the *Grundy double domination number*, denoted by $\gamma_{2gr}(G)$. A double dominating sequence of length $\gamma_{2gr}(G)$ is called a γ_{2gr} -sequence or a *Grundy double dominating sequence*.

Note that if S is a double neighborhood sequence according to our definition then \widehat{S} is a double dominating set of G . But according to the definition 1, if S is a double neighborhood sequence of G then \widehat{S} is not necessarily a double dominating set of G . If we consider a maximum length double neighborhood sequence according to the definition 1, then the set \widehat{S} is a double dominating set of G . Hence, optimal solutions according to both definitions, will be same. Below, we prove it formally.

Before the proof, we define some notations that will be used throughout this chapter. If $S_1 = (v_1, v_2, \dots, v_n)$ and $S_2 = (u_1, u_2, \dots, u_m)$, $n, m \geq 0$, are two sequences, then the *concatenation* of the sequences S_1 and S_2 is the sequence $S_1 \oplus S_2 = (v_1, v_2, \dots, v_n, u_1, u_2, \dots, u_m)$. Let $A = \{u_1, u_2, \dots, u_t\}$ be an ordered set of vertices then (A) denotes the sequence of vertices (u_1, u_2, \dots, u_t) . For vertices u_1, u_2, \dots, u_t , the sequence (u_1, u_2, \dots, u_t) is denoted by $u_1 \oplus u_2 \oplus \dots \oplus u_t$.

Let S be a double neighborhood sequence according to the definition 1. We see that S is a double neighborhood sequence according to our definition also. Now, either \widehat{S} is a double dominating set of G or after appending some vertices at the end of S , we get a double dominating sequence of G according to our definition. So, we state the following lemma.

Lemma 4.1. $\gamma_{2gr}(G) \leq \gamma_{gr}^{\times 2}(G)$.

We fix some terminologies for this section only. In a double neighborhood sequence S according to our definition, a vertex $u \in \widehat{S}$ is said to be a *good vertex* if it is appearing in the sequence only to double dominate vertices appearing prior to u .

In order to prove that the value of Grundy double domination number is equal in all graphs according to both the definitions, it is enough to prove the following lemma.

Lemma 4.2. *There exists a double neighborhood sequence according to definition 1 of size $\gamma_{gr}^{\times 2}(G)$.*

Proof. Consider a Grundy double dominating sequence $S = (v_1, v_2, \dots, v_k)$ of G according to our definition, in which the index of the first good vertex is highest. Let i be the index of the first good vertex in S . We write S as $S_1 \oplus v_i \oplus S_2$. Since v_i is a good vertex, it dominates some vertices of the graph the second time and all of those vertices appear before v_i in the sequence. Assume that v_i dominates vertices $v_{i_1}, v_{i_2}, \dots, v_{i_r}$ second time and they appear before v_i in S in the order $v_{i_1}, v_{i_2}, \dots, v_{i_r}$. Note that these vertices appear in the subsequence S_1 . Since v_i dominates the vertices $v_{i_1}, v_{i_2}, \dots, v_{i_r}$ second time, we get that no neighbor of these vertices appears in the subsequence S_1 . Now, we modify S as follows. We place vertices $v_{i_1}, v_{i_2}, \dots, v_{i_r}$ just after v_i in the same order and get a new sequence, say S' .

Next, we show that there is no good vertex till index i in S' and S' remains a double neighborhood sequence of G according to our definition. Clearly, the vertex at the i -th index in S' is v_{i_r} and it dominates itself the second time. Note that S' contains no good vertex up to the vertex v_{i_r} . So, there is no good vertex till the index i in S' . Now, it remains to show that S' is still a double neighborhood sequence of G according to our definition of size k . We write S' as $S'_1 \oplus v_i \oplus v_{i_1} \oplus v_{i_2} \oplus \dots \oplus v_{i_r} \oplus S'_2$. Note that S'_2 is same as the subsequence S_2 .

Let $x \in \widehat{S}'_1$. Since S was a double neighborhood sequence of G according to our definition, x dominates a vertex of G which is dominated at most once by the vertices prior to x in S . Let A be the set of vertices that appear before x in S . In the sequence S' , vertex x is appearing before v_i and the set of vertices that appear prior to x in S' is a subset of A . So, x still dominates a vertex which is dominated at most once before x in S' . Now, we see that v_i dominates vertices $v_{i_1}, v_{i_2}, \dots, v_{i_r}$ first time which are appearing after v_i in S' . Vertices $v_{i_1}, v_{i_2}, \dots, v_{i_r}$ dominate themselves second time in S' . So, every vertex of $S'_1 \cup \{v_i, v_{i_1}, v_{i_2}, \dots, v_{i_r}\}$ dominates a vertex which is dominated at most once by the previous vertices. Finally, let $x \in S'_2$. Clearly x dominates a vertex which is dominated at most once by the vertices which precede x in S' .

Hence, S' remains a double neighborhood sequence of size $|\widehat{S}|$ according to our definition and so, S' is also a Grundy double dominating sequence of G according to our definition. The sequence S' contains no good vertex till index i . This contradicts the fact that S is a Grundy double dominating sequence according to our definition in which the first good vertex appears at the highest index. Thus, there is a sequence that contains no good

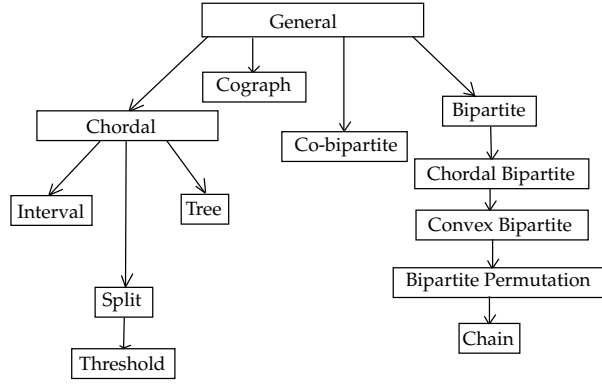


FIGURE 4.1: Some Graph Classes.

vertex. Therefore, there exists a double neighborhood sequence according to definition 1 of size $\gamma_{gr}^{\times 2}(G)$. \square

Using Lemma 4.1 and Lemma 4.2, we get that $\gamma_{2gr}(G) = \gamma_{gr}^{\times 2}(G)$. **Therefore, we follow our definition of double dominating sequences in this thesis.**

A hierarchy presenting relationships between some classes of graphs that are relevant to this chapter is shown in Fig. 4.1. The section-wise organization of this chapter is as follows: In Section 4.2, we present all the results we have obtained for the GD problem, and Section 4.3 describes all the results concerning the GD2 problem.

4.2 Dominating Sequences

In this section, we first prove that the GDD problem is NP-complete for bipartite and co-bipartite graphs. Next, we propose a linear-time algorithm that outputs a Grundy dominating sequence for a chain graph, a subclass of bipartite graphs.

In the seminal paper [20] the authors proved that the GDD problem is NP-complete for chordal graphs. They also proved that a Grundy dominating sequence in trees, cographs, and split graphs can be computed in polynomial time [20]. An additional study of Grundy domination in forests was used in proving a formula for the Grundy domination number in strong products of graphs [12]. Several combinatorial results have also been established for the parameter and its relatives in the literature [18, 24, 34, 44, 63]. From the computational point of view, it was shown that the GDD problem can be solved in polynomial time for

interval graphs and Sierpiński graphs [19], as well as on some X -join products, lexicographic products and related classes of graphs [72].

If S is a closed neighborhood sequence, then we say that v_i *footprints* the vertices from $N[v_i] \setminus \cup_{j=1}^{i-1} N[v_j]$, and that v_i is the *footprinter* of every vertex $u \in N[v_i] \setminus \cup_{j=1}^{i-1} N[v_j]$.

The following result is an immediate consequence of definitions.

Proposition 10. Let S be a (Grundy) dominating sequence in a graph G . If $u, v \in V(G)$ such that $N[u] \subseteq N[v]$ and $u, v \in \widehat{S}$, then u appears before v in S .

Given a hypergraph $H = (X, \mathcal{E})$, a *legal transversal sequence* is a sequence $S = (v_1, \dots, v_k)$ of vertices from X such that for each i there exists an hyperedge $\mathcal{E}_i \in \mathcal{E}$ such that $v_i \in \mathcal{E}_i$ and $v_j \notin \mathcal{E}_i$ for all j , where $j < i$. The longest possible legal transversal sequence in a hypergraph H is a *Grundy transversal sequence* and its length is the *Grundy transversal number* of H , denoted $\tau_{gr}(H)$. The following result was proved in [22, Proposition 8.3].

Proposition 11. The Grundy transversal number of an arbitrary hypergraph H equals the Grundy covering number of H ; $\tau_{gr}(H) = \rho_{gr}(H)$.

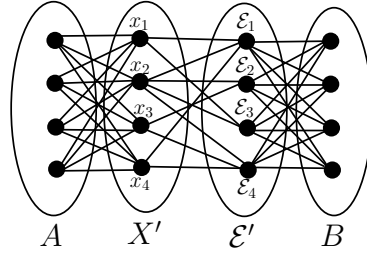
4.2.1 NP-completeness results

Recall that the GDD problem is NP-complete for general graphs, and also when restricted to chordal graphs. In this subsection, we prove that the problem remains NP-complete for bipartite and co-bipartite graphs.

4.2.1.1 Bipartite Graphs

Here, we prove the NP-completeness of the GDD problem for bipartite graphs. We reduce the GCD problem for hypergraphs to the GDD problem for bipartite graphs. Given a hypergraph $H = (X, \mathcal{E})$ with $|X| = n$ and $|\mathcal{E}| = m$, ($n, m \geq 2$), we construct an instance $G = (X^*, Y^*, E^*)$ of the GDD problem, where G is a bipartite graph, as follows.

$V(G) = A \cup X' \cup \mathcal{E}' \cup B$, where $A = \{a_1, a_2, \dots, a_n\}$ and $B = \{b_1, b_2, \dots, b_m\}$. The sets X' and \mathcal{E}' contain n and m vertices respectively, where each vertex of X' corresponds to a vertex of X in the hypergraph H and each vertex of \mathcal{E}' correspond to a hyperedge of H .

FIGURE 4.2: Construction of bipartite graph G from the hypergraph H .

For a hyperedge $\mathcal{E}_i \in \mathcal{E}$, we denote the corresponding vertex in \mathcal{E}' by e_i . Now, a vertex x of X' is adjacent to a vertex of $e_i \in \mathcal{E}'$ in G if and only if $x \in \mathcal{E}_i$ in H . Each vertex of A is adjacent to each vertex of X' in G , also, each vertex of B is adjacent to each vertex of \mathcal{E}' in G . Clearly, graph G is a bipartite graph. Fig. 4.2 illustrates the construction of G when H is the hypergraph given by $(X = \{x_1, x_2, x_3, x_4\}, \mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4\})$, where $\mathcal{E}_1 = \{x_1, x_2, x_4\}$, $\mathcal{E}_2 = \{x_2, x_3\}$, $\mathcal{E}_3 = \{x_1, x_2\}$ and $\mathcal{E}_4 = \{x_2, x_3, x_4\}$.

Theorem 4.3. *Let G be the bipartite graph constructed from a hypergraph $H = (X, \mathcal{E})$ with $|X| = n$ and $|\mathcal{E}| = m$, ($n, m \geq 2$) as explained above. Then, $\rho_{gr}(H) \geq k$ if and only if $\gamma_{gr}(G) \geq n + m + k$.*

Proof. First, let $(\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_{k'})$ be an edge covering sequence of size at least k in H . Then the sequence $(b_1, b_2, \dots, b_m, e_1, e_2, \dots, e_{k'}, a_1, a_2, \dots, a_n)$ is a dominating sequence of size at least $n + m + k$ in G . Hence, $\gamma_{gr}(G) \geq n + m + k$.

For the converse, let us assume that $\gamma_{gr}(G) \geq n + m + k$ for some positive integer k . If $X' \cap \widehat{S} \neq \emptyset$ for some dominating sequence S of G , then x_0 denotes the first vertex in S coming from X' , and if $\mathcal{E}' \cap \widehat{S} \neq \emptyset$ for some dominating sequence S of G , then e_0 denotes the first vertex in S coming from \mathcal{E}' .

First, we prove two auxiliary claims.

Claim 1. There exists a dominating sequence S of size at least $n + m + k$ in G such that all vertices of A appear in S and if $X' \cap \widehat{S} \neq \emptyset$, then all vertices of A appear before x_0 .

Proof. Let S be a dominating sequence of size at least $n + m + k$ in G (which exists, since $\gamma_{gr}(G) \geq n + m + k$). Suppose there exists a vertex $a_i \in A$, which is not appearing in S .

Then, there exists vertex from X' which is appearing in S to footprint a_i . Hence, $X' \cap \widehat{S} \neq \emptyset$ and x_0 footprints a_i . Let P denotes the set of vertices appearing before x_0 and Q be the set of vertices appearing after x_0 in S . Now, two cases are possible.

Case 1: $Q \cap A = \emptyset$.

In this case, either $P \cap A = \emptyset$ or P contains some vertices of A . So, first assume that P contains no vertex of A . Then, we see that no vertex of A appears in the sequence S . If x_0 does not footprint any vertex in \mathcal{E}' , then we modify S by appending vertices of A in the order (a_1, a_2, \dots, a_n) just before x_0 and removing all vertices from Q that footprinted a vertex of X' along with the vertex x_0 . Otherwise, if x_0 footprints some vertices of \mathcal{E}' , we perform the same modification without removing x_0 . In either case, we removed at most n vertices and we added n new vertices to S , by which the so modified sequence S is a dominating sequence of size at least $n + m + k$ in G , which satisfies the statement of the claim.

Now, if P contains some vertices of A , then no vertex of Q footprints any vertex of X' . Again, if x_0 footprints only vertices of A , then we modify S by appending vertices of $A \setminus P$ in any order just before x_0 and removing the vertex x_0 . Otherwise, if x_0 footprints also some vertices of \mathcal{E}' , we perform the same modification, but without removing x_0 . In either case, we removed at most 1 vertex and we added at least 1 new vertex to the sequence S . With this, the so modified sequence S is a dominating sequence of size at least $n + m + k$ in G , which satisfies the statement of the claim.

Case 2: $|Q \cap A| = 1$.

In this case, P contains no vertex of A . Let a_j be the vertex from $Q \cap A$ appearing in S . Note that the vertices in S , which footprint only vertices from X' , do not appear after a_j in S . Note that there are at most $n - 2$ vertices that appear in S between x_0 and a_j and footprint a vertex of X' , and denote the set of these vertices by Q' . If x_0 does not footprint any vertex in \mathcal{E}' , then we modify S by appending vertices of A in the order (a_1, a_2, \dots, a_n) just before x_0 and then removing all vertices of $Q' \cup \{x_0, a_j\}$. Otherwise, if x_0 footprints also some vertices of \mathcal{E}' , we perform the same modification without removing x_0 . In either case, we removed at most $n - 1$ vertices and we added $n - 1$ new vertices to S . Hence, the so modified sequence S is a dominating sequence of size at least $n + m + k$ in G , which satisfies the statement of the claim. \square

By the above claim, there exists a dominating sequence S of size $n + m + k$ such that all vertices of A appear in S and they appear before x_0 if $X' \cap \widehat{S} \neq \emptyset$. The proof of Claim 2 is similar to the proof of the Claim 1.

Claim 2. There exists a dominating sequence S of size at least $n + m + k$ in G such that all vertices of B appear in S and if $\mathcal{E}' \cap \widehat{S} \neq \emptyset$, then all vertices of B appear before e_0 .

Combining the above two claims we infer that there exists a dominating sequence S of size at least $n + m + k$ in G such that $|(X' \cup \mathcal{E}') \cap \widehat{S}| \geq k$.

Claim 3. Either $X' \cap \widehat{S} = \emptyset$ or $\mathcal{E}' \cap \widehat{S} = \emptyset$.

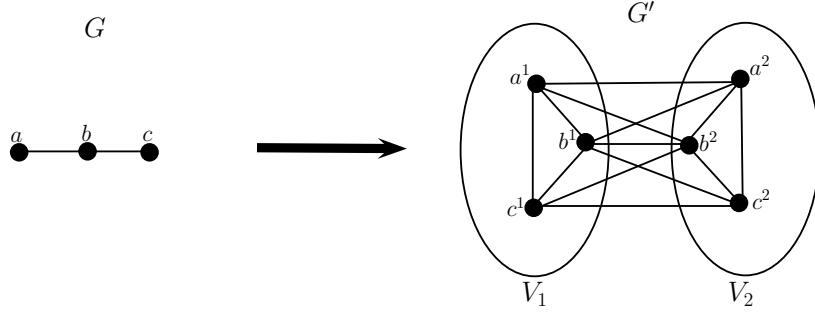
Proof. If $X' \cap \widehat{S} = \emptyset$, we are done. So, assume that $X' \cap \widehat{S} \neq \emptyset$ and $\mathcal{E}' \cap \widehat{S} \neq \emptyset$. Now, either e_0 appears before x_0 or e_0 appears after x_0 . In the former case, we see that before the vertex x_0 , all vertices of G are footprinted (and thus dominated) using Claims 1 and 2. So, x_0 does not footprint any vertex implying that this case is not possible. Similarly, the latter case is also not possible, which proves the claim. \square

Now, if $X' \cap \widehat{S} = \emptyset$, then we have that $|\mathcal{E}' \cap \widehat{S}| \geq k$. In addition, by Claim 2, since all vertices of B appear in S before any vertex of \mathcal{E}' appears in S , the subsequence of S of vertices in \mathcal{E}' corresponds to an edge covering sequence in the hypergraph H , which is of size at least k . Thus, $\rho_{gr}(H) \geq k$, as desired.

Otherwise, if $\mathcal{E}' \cap \widehat{S} = \emptyset$, then we derive that $|X' \cap \widehat{S}| \geq k$, where the subsequence formed by vertices of $X' \cap \widehat{S}$ corresponds to a legal transversal sequence of the hypergraph H of size at least k . By Proposition 11, $\tau_{gr}(H) = \rho_{gr}(H)$, and so $\rho_{gr}(H) \geq k$. The proof of the theorem is complete. \square

The following theorem follows directly from Theorem 4.3.

Theorem 4.4. *The GDD problem is NP-complete for bipartite graphs.*

FIGURE 4.3: Construction of a co-bipartite graph G' from a graph G .

4.2.1.2 Co-bipartite Graphs

Now, we prove the NP-completeness of the GDD problem for co-bipartite graphs. Here, we reduce the GDD problem for general graphs to the GDD problem for co-bipartite graphs. Given a graph $G = (V, E)$, where $V = \{v_1, v_2, \dots, v_n\}$, we construct an instance $G' = (V_1 \cup V_2, E')$ of the GDD problem, where G' is a co-bipartite graph, as follows.

The vertex set of G' is $V_1 \cup V_2$, where $V_1 = \{v_i^1 : v_i \in V\}$ and $V_2 = \{v_i^2 : v_i \in V\}$. The set of edges of G' is given by $\{v_i^1 v_j^1 : 1 \leq i < j \leq n\} \cup \{v_i^2 v_j^2 : 1 \leq i < j \leq n\} \cup \{v_i^1 v_j^2 : v_j \in N_G[v_i], i, j \in [n]\}$. Note that G' is a co-bipartite graph. Fig. 4.3 provides an illustration of the construction of G' from G .

Claim 4. For a positive integer k , $\gamma_{gr}(G) \geq k$ if and only if $\gamma_{gr}(G') \geq k$.

Proof. First, let $S = (u_1, u_2, \dots, u_t)$ be a dominating sequence of G of size t , where $t \geq k$. Then $S' = (u_1^1, u_2^1, \dots, u_t^1)$ is a dominating sequence of size at least k in G' . Indeed, if v_i is a vertex footprintd by u_i with respect to S , then v_i^2 is footprintd by u_i^1 with respect to S' .

Conversely, let $S = (w_1, w_2, \dots, w_t)$ be a dominating sequence of size t in G' , where $t \geq k$ and $w_i \in V_1 \cup V_2$ for all $i \in [t]$. Without loss of generality, we may assume that $w_1 \in V_1$. Note that there can be at most one vertex from V_2 in S . If there is no such vertex, then the sequence S corresponds to a sequence of vertices in G of size at least k , which is a dominating sequence of G . Now, suppose there exists a vertex from V_2 in S . Clearly, it has to be the last vertex of S , and let w_t be vertex $v_i^2 \in V_2$. Note that v_i^2 appears in S to footprint some vertices of V_2 . Let $K = \{v_j^2 \in V_2 : v_j^2 \text{ is footprintd by } v_i^2 \text{ in } S\}$. Let $v_j^2 \in K$ then, $v_j^1 \notin \widehat{S}$. We modify S by replacing the vertex v_i^2 with the vertex v_j^1 and get a new sequence

S' . If S' is not a dominating sequence then there is a vertex $v_r^2 \in K$ which is not dominated by the sequence S' . In this case, we see that the sequence $S' \oplus v_r^1$ dominates v_r^2 . We keep on appending such vertices until every vertex of K is dominated and call the final sequence again by S' . So, the updated S' is a dominating sequence of G' of size at least k . Since S' contains only vertices from V_1 , it corresponds to a sequence of vertices in G of size at least k , which is a dominating sequence of G . This completes the proof of the converse direction of the statement. \square

Now, we are ready to state the announced result.

Theorem 4.5. *The GDD problem is NP-complete for co-bipartite graphs.*

4.2.2 Algorithm for Chain Graphs

In this subsection, we give a linear-time algorithm to compute a GD-sequence of a chain graph. Before discussing the main idea for chain graphs, we first give the Grundy domination number of a complete bipartite graph which is a subclass of chain graphs. The proof of this result is easy and hence, is omitted.

Proposition 12. If $G = (X, Y, E)$ is complete bipartite graph, then $\gamma_{gr}(G) = \max\{|X|, |Y|\}$.

A chain graph $G = (X, Y, E)$ has a chain ordering (O_X, O_Y) , where $O_X = (x_1, x_2, \dots, x_{n_1})$ and $O_Y = (y_1, y_2, \dots, y_{n_2})$, and based on the equivalence relation \sim discussed in Chapter 1, the sets X and Y have a twin partition. Let $P_X = \{X_1, X_2, \dots, X_k\}$ be the twin partition of the vertex subset X and $P_Y = \{Y_1, Y_2, \dots, Y_k\}$ be the twin partition of the vertex subset Y . Recall that $N(X_i) = \bigcup_{r=1}^i Y_r$ and $N(Y_j) = \bigcup_{r=j}^k X_r$ for each $i, j \in [k]$. Since the case $k = 1$ yields a complete bipartite graph, in the rest of this subsection, we only consider the chain graphs with $k \geq 2$. We also assume that a chain graph $G = (X, Y, E)$ is given along with the chain ordering and the twin partitions of X and Y .

The proof of the following observation is again easy, and hence is omitted.

Observation 4. Let $A \subseteq V(G)$ be a set of open twins in an arbitrary graph G . Then there exists a GD-sequence S of G such that all vertices of $A \cap \hat{S}$ appear together in S .

Observation 5. Let S be a GD-sequence of G and $A \subseteq V(G)$ be a set of open twins in an arbitrary graph G such that $A \cap \hat{S} \neq \emptyset$. If the first vertex of A in S footprints itself, then there exists a GD-sequence of G in which all vertices of A appear and they appear together in that sequence.

Proof. If $A \cap \hat{S} = A$, we have nothing to prove. So, suppose that $A \cap \hat{S} \neq A$. Then, there exists a vertex $a \in A$, which is not in S . Thus there exists a vertex $b \in \hat{S}$, which footprints a . Note that b appears after all vertices of $A \cap \hat{S}$. Now, we modify S by replacing b with the vertex a . By doing this repeatedly, we get a new GD-sequence S' of G which contains all vertices of A . We can rearrange all vertices of A so that all vertices of A appear together in S' . \square

In the remainder of the subsection we assume that G is a chain graph with the partition of its vertex set as described earlier.

Observation 6. There exists a GD-sequence S of G such that for every $i \in [k]$, we have $X_i \cap \hat{S} \neq \emptyset$ or $Y_i \cap \hat{S} \neq \emptyset$.

Proof. Let S be a GD-sequence such that $X_i \cap \hat{S} = \emptyset$ and $Y_i \cap \hat{S} = \emptyset$. This implies that there exist vertices $x \in \cup_{r=i+1}^k X_r$ and $y \in \cup_{r=1}^{i-1} Y_r$ in S to footprint the vertices of Y_i and X_i respectively. Note that $i \notin \{1, k\}$. Now, if x appears before y in S , then modifying the sequence S by replacing y by all vertices of X_i gives another GD-sequence of G such that $X_i \cap \hat{S} \neq \emptyset$. (Note that if $y \in Y_j$, then $X_j \cap \hat{S} \neq \emptyset$. To see this, assuming that $X_j \cap \hat{S} = \emptyset$ implies that y footprints vertices of both X_j and X_i in S . In this case, replacing the vertex y with all vertices of X_i and X_j results in a new dominating sequence of G of length bigger than S , a contradiction.) Similarly, if y appears before x in S , then modifying the sequence S by replacing x by all vertices of Y_i gives another GD-sequence of G such that $Y_i \cap \hat{S} \neq \emptyset$. Hence, there exists a GD-sequence S of G such that $X_i \cap \hat{S} \neq \emptyset$ or $Y_i \cap \hat{S} \neq \emptyset$. \square

Let A be a set of open twins in G . If $|A \cap \hat{S}| \geq 2$, then we see that the first vertex of A in S footprints itself. Thus we can assume that, $A \subseteq \hat{S}$, due to Observation 5. Hence, we

have, $|A \cap \hat{S}| \leq 1$ or $A \subseteq \hat{S}$ for any set of open twins A in G . Note that the each of the sets $X_1, X_2, \dots, X_k, Y_1, Y_2, \dots, Y_k$ is a set of open twins in G .

Now, based on the Observations 4, 5 and 6, whenever we consider a GD-sequence S of G , we assume that S satisfies the following, for the rest of this section:

- (1) For each $i \in [k]$, $|X_i \cap \hat{S}| \leq 1$ or $X_i \subseteq \hat{S}$. If $X_i \subseteq \hat{S}$, then all vertices of X_i appear together in S .
- (2) For each $i \in [k]$, $|Y_i \cap \hat{S}| \leq 1$ or $Y_i \subseteq \hat{S}$. If $Y_i \subseteq \hat{S}$, then all vertices of Y_i appear together in S .
- (3) For each $i \in [k]$, $X_i \cap \hat{S} \neq \emptyset$ or $Y_i \cap \hat{S} \neq \emptyset$.

Now, let S be a GD-sequence of G . Then S is one of the following type:

$$(a) X \cap \hat{S} = \emptyset, \quad (b) Y \cap \hat{S} = \emptyset, \quad (c) X \cap \hat{S} \neq \emptyset \text{ and } Y \cap \hat{S} \neq \emptyset.$$

We call the corresponding GD-sequences S to be of *type (a)*, *type (b)*, and *type (c)*, respectively.

Lemma 4.6. *Let $S^* = (v_1, v_2, \dots, v_p)$ be a GD-sequence of G of type (c). The following statements hold:*

- 1. *If $v_1 \in Y_1$, then there exists a type (a) GD-sequence of G .*
- 2. *If $v_1 \in X_k$, then there exists a type (b) GD-sequence of G .*
- 3. *If $v_1 \notin Y_1 \cup X_k$, then there exists a GD-sequence S of G such that $\cup_{r=1}^i X_r \subseteq \hat{S}$ for some $i \in [k]$ and $\cup_{r=j}^k Y_r \subseteq \hat{S}$ for some $j \in [k]$.*

Proof. First, we assume that $v_1 \in Y_1$. In this case, all vertices of X are footprinted by v_1 . So, all the vertices v_2, \dots, v_p appear to footprint vertices of $Y \setminus \{v_1\}$ only. This implies that $\gamma_{gr}(G) \leq |Y|$. So, the sequence $S = (y_{n_2}, y_{n_2-1}, \dots, y_1)$ is also a GD-sequence of G and it is of type (a).

Next, we assume that $v_1 \in X_k$. In this case, all vertices of Y are footprinted by v_1 . So, all the vertices v_2, \dots, v_p appear to footprint vertices of $X \setminus \{v_1\}$ only. This implies that $\gamma_{gr}(G) \leq |X|$. So, the sequence $S = (x_1, x_2, \dots, x_{n_1})$ is also a GD-sequence of G and it is of type (b).

Next, we assume that $v_1 \notin Y_1 \cup X_k$. Since S^* contains vertices from both X and Y , we have two cases to consider.

Case 1: $v_1 \in X$.

Let $v_1 \in X_{i_0}$ ($i_0 \in [k]$). So, we get that v_1 footprints all vertices of $N(X_{i_0}) \cup \{v_1\}$. Now, let u be a vertex of X such that $N(u) \subseteq N(v_1)$. If u is footprinted by some vertex from $N(u)$, we modify the sequence S^* as follows. We remove the footer of u from S^* and include u just after v_1 and get a new sequence. But, if u is footprinted by itself, then we relocate u in S by putting it just after v_1 . We repeat the respective modifications for each vertex u such that $N(u) \subseteq N(v_1)$ and get a new sequence S which remains a GD-sequence of G . Thus, we have that $\cup_{r=1}^{i_0} X_r \subseteq \hat{S}$ and the vertices of $\cup_{r=1}^{i_0} X_r$ are at first $\sum_{r=1}^{i_0} |X_r|$ places of the sequence. Next, we consider the $(\sum_{r=1}^{i_0} |X_r| + 1)$ -th vertex of the sequence. If that vertex is from X , we repeat the same modifications until we get a vertex of Y in the sequence. After all such modifications, we have that $\cup_{r=1}^i X_r \subseteq \hat{S}$ for some $i \in [k]$.

Now, we again rename the sequence by S^* and vertices of S^* by (v_1, v_2, \dots, v_p) . Let $t \in [p]$ be the smallest index such that $v_t \in Y$. Note that all vertices of $\cup_{r=1}^i (X_r \cup Y_r)$ are footprinted before the appearance of v_t . If $v_t \in \cup_{r=1}^i Y_r$, then v_t footprints all vertices of $\cup_{r=i+1}^k X_r$. With this, all vertices of X are footprinted. So, a vertex $v_{t'} \in \hat{S}^*$ such that $t' > t$ appears to footprint a vertex of $\cup_{r=i+1}^k Y_r$ and no vertex of $\cup_{r=i+1}^k Y_r$ appeared before v_t . We modify S^* by removing all vertices of S^* after v_t and including all vertices of $\cup_{r=i+1}^k Y_r$ after v_t , we get a new GD-sequence S in which we have $\cup_{r=i+1}^k Y_r \subseteq \hat{S}$. Otherwise, if $v_t \in Y_{i'}$ where $i' > i$, then v_t footprints all vertices of $\cup_{r=i'}^k X_r$. Clearly, all vertices of $\cup_{r=i'}^k Y_r$ need to be footprinted after the appearance of v_t in the sequence S^* . If some vertex of $\cup_{r=i'}^k Y_r$ is not appearing in S^* then we remove its footer from the sequence and include the vertex itself in the sequence. Thus, we get a new GD-sequence S in which we have $\cup_{r=i'}^k Y_r \subseteq \hat{S}$.

Case 2: $v_1 \in Y$.

Similar proof as case 1.

Therefore, there exists a GD-sequence S of G such that $\cup_{r=1}^i X_r \subseteq \hat{S}$ for some $i \in [k]$ and $\cup_{r=j}^k Y_r \subseteq \hat{S}$ for some $j \in [k]$. \square

Lemma 4.7. *Suppose there exists a type (c) GD-sequence (v_1, v_2, \dots, v_p) of G , where $v_1 \notin (X_1 \cup Y_k)$. Then there exists a GD-sequence S of G such that $\cup_{r=1}^i X_r \subseteq \hat{S}$ for some $i \in [k]$ and $\cup_{r=j}^k Y_r \subseteq \hat{S}$ for some $j \in [k]$, where $j \geq i$.*

Proof. Using Lemma 4.6, we have that there exists a GD-sequence S of G such that $\cup_{r=1}^i X_r \subseteq \hat{S}$ for some $i \in [k]$ and $\cup_{r=j}^k Y_r \subseteq \hat{S}$ for some $j \in [k]$. We need to show that $j \geq i$. On the contrary, assume that, $j < i$, then we have, $(X_{i-1} \cup X_i \cup Y_{i-1} \cup Y_i) \subseteq \hat{S}$. Let $\mathcal{K} = \{X_{i-1}, X_i, Y_{i-1}, Y_i\}$. Recall that all vertices coming from a set of open twins appear together in S . Let $A \in \mathcal{K}$ be the set whose vertices appear after the other three sets of \mathcal{K} in the sequence S . Then, all vertices of $N[A]$ are footprinted before the appearance of vertices of A , so A does not footprint a new vertex, a contradiction. Therefore, $j \geq i$. \square

Lemma 4.8. *Suppose there exists a type (c) GD-sequence (v_1, v_2, \dots, v_p) of G , where $v_1 \notin (X_1 \cup Y_k)$. Then there exists a GD-sequence S of G such that $\cup_{r=1}^i X_r \subseteq \hat{S}$ for some $i \in [k]$ and $\cup_{r=j}^k Y_r \subseteq \hat{S}$ for some $j \in [k]$, where $j \geq i$. Moreover, the following is satisfied by the sequence S :*

1. *If $i \leq k - 2$, either $(\cup_{r=i+1}^k X_r) \cap \hat{S} = \emptyset$ or $(\cup_{r=i+2}^k X_r) \cap \hat{S} = \emptyset$ and $|X_{i+1} \cap \hat{S}| = 1$.*
2. *If $i = k - 1$, then either $X_k \cap \hat{S} = \emptyset$ or $|X_k \cap \hat{S}| = 1$.*

Proof. Let S be a GD-sequence of G such that $\cup_{r=1}^i X_r \subseteq \hat{S}$ for some $i \in [k]$ and $\cup_{r=j}^k Y_r \subseteq \hat{S}$ for some $j \in [k]$, where $j \geq i$. We also assume that S is a sequence with largest such i . Existence of such a sequence is ensured by the Lemmas 4.6 and 4.7.

Now, assume that $i < k$ and so, $X_{i+1} \not\subseteq \hat{S}$. If $i = k - 1$ and $X_k \cap \hat{S} \neq \emptyset$ then either $|X_k \cap \hat{S}| = 1$ or $|X_k \cap \hat{S}| \geq 2$. In the latter case, we get that first vertex of X_k in S footprints itself. So, Observation 5 ensures that $|X_k \cap \hat{S}| = 1$.

Next, we show that if $i \leq k - 2$, then $(\cup_{r=i+2}^k X_r) \cap \hat{S} = \emptyset$. So, let $t \in \{i + 2, \dots, k\}$ be the minimum index such that $X_t \cap \hat{S} \neq \emptyset$. This means that there are some vertices of $\cup_{r=i+1}^{t-1} X_r$ which are not appearing in the sequence S . Let A denotes the set of these vertices. Note that A is not the empty set. Let x be the vertex of X_t which appears first in S . We discuss two cases here.

Case 1: x footprints itself.

In this case, we have that no neighbor of X_t appears in S before x . So, all vertices in S , which footprint vertices of A , appear after x . Now, we modify S by removing all such vertices and including all vertices of A in the sequence just after all vertices of $X_t \cap \hat{S}$. We call the modified sequence again by S as it remains a GD-sequence of G . Thus, we get a contradiction on i being the largest index. So, this case is not possible.

Case 2: x does not footprint itself.

In this case, we get that all vertices of X_t are footprinted by some vertex of Y , which appears before x in S . So, x footprints some vertices from the set $\cup_{r=1}^t Y_r$. Thus, $|X_t \cap \hat{S}| = 1$ and $X_t \cap \hat{S} = \{x\}$. Now, let y be the vertex which footprints vertices of $A \cap X_{i+1}$ in S . Then, there can be two subcases:

Subcase 2.1: y appears after x .

In this subcase, we modify S by removing y and including all vertices of $A \cap X_{i+1}$ in the sequence just after x . We call the modified sequence again by S as it remains a GD-sequence of G . Thus, we again get a contradiction on i being the largest index. So, this subcase is not possible.

Subcase 2.2: y appears before x .

Here, all vertices of A are footprinted before the appearance of x . Recall that $x \in (\cup_{r=i+2}^k X_r) \cap \hat{S}$. We get that all vertices of $\cup_{r=i+1}^k X_r$ are footprinted before the appearance of x . Note that the vertex x itself is footprinted before the appearance of x . So, we have, x appears to footprint some vertices of $\cup_{r=1}^t Y_r$. This can be further divided in two cases: (i) x does not footprint the vertices of Y_t . (ii) x footprints the vertices of Y_t .

In the first case, x footprints some vertices of the set $\cup_{r=1}^{t-1} Y_r$. Note that $A \cap X_{t-1} \neq \emptyset$

and vertices of $A \cap X_{t-1}$ does not appear in S . Now, we modify S by replacing the vertex x by a vertex of $A \cap X_{t-1}$ and get a new GD-sequence in which no vertex of X_t appears and one vertex of X_{t-1} appears. If $t = i + 2$, then after applying the modification once, we get a GD-sequence S' such that $X_{i+2} \cap \hat{S}' = \emptyset$ and $|X_{i+1} \cap \hat{S}'| = 1$. Otherwise, if $t > i + 2$, then after applying the modification once, we get a GD-sequence S' such that $X_t \cap \hat{S}' = \emptyset$ and $|X_{t-1} \cap \hat{S}'| = 1$. Thus, we have another index $t' = t - 1 \in \{i + 2, \dots, k\}$ such that $X_{t'} \cap \hat{S}' \neq \emptyset$. Now, we consider the vertex of $X_{t-1} \cap \hat{S}'$. Clearly, the vertex of $X_{t-1} \cap \hat{S}'$ also footprints some neighbor of it. We repeat the similar modifications until we get a GD-sequence in which a vertex of X_p , where $p \in \{i + 2, \dots, k\}$ appears in the sequence and it appears for a vertex of Y_p . When we arrive at such a sequence, we perform one more modification by replacing the vertex of X_p by vertices of Y_p .

In the second case, x footprints vertices of Y_t and we modify S by replacing x with all vertices of Y_t . Note that no vertex of Y_t was appearing in the sequence prior to this modification. If $t = i + 2$, then after applying the modification once, we get a GD-sequence S' such that $X_{i+2} \cap \hat{S}' = \emptyset$. Otherwise, if $t > i + 2$, then after applying the modification once, we get a GD-sequence S' such that $X_t \cap \hat{S}' = \emptyset$ and $X_{t-1} \cap \hat{S}' = \emptyset$. Now, we have, either there is no index t' in the set $\{i + 2, \dots, k\}$ such that $X_{t'} \cap \hat{S}' \neq \emptyset$ or there is some $t' \in \{i + 2, \dots, k\}$ ($t' > t$) such that $X_{t'} \cap \hat{S}' \neq \emptyset$.

By repeating the above arguments we get that, there is a GD-sequence S of G such that either $(\cup_{r=i+1}^k X_r) \cap \hat{S} = \emptyset$ or $(\cup_{r=i+2}^k X_r) \cap \hat{S} = \emptyset$ and $|X_{i+1} \cap \hat{S}| = 1$. Note that during the modifications, whenever we removed a vertex from the sequence, it was a vertex of $\cup_{r=i+2}^k X_r$ so, $\cup_{r=1}^i X_r$ remains a subset of \hat{S} and whenever we added a vertex to the sequence, it was a vertex of Y_r , where a vertex of Y_r was missing from the sequence prior to any modification, implying $r < j$ so, $\cup_{r=j}^k Y_r$ remains a subset of \hat{S} . Therefore, the Lemma holds. \square

Lemma 4.9. *Suppose there exists a type (c) GD-sequence (v_1, v_2, \dots, v_p) of G , where $v_1 \notin (X_1 \cup Y_k)$. Then there exists a GD-sequence S of G such that $\cup_{r=1}^i X_r \subseteq \hat{S}$ for some $i \in [k]$ and $\cup_{r=j}^k Y_r \subseteq \hat{S}$ for some $j \in [k]$, where $j \geq i$. Moreover, the following is satisfied by the sequence S :*

1. If $i \leq k - 2$, either $(\cup_{r=i+1}^k X_r) \cap \hat{S} = \emptyset$ or $(\cup_{r=i+2}^k X_r) \cap \hat{S} = \emptyset$ and $|X_{i+1} \cap \hat{S}| = 1$.
2. If $i = k - 1$, then either $X_k \cap \hat{S} = \emptyset$ or $|X_k \cap \hat{S}| = 1$.
3. $j \leq i + 2$.
4. If $j \geq 3$, either $(\cup_{r=1}^{j-1} Y_r) \cap \hat{S} = \emptyset$ or $(\cup_{r=1}^{j-2} Y_r) \cap \hat{S} = \emptyset$ and $|Y_{j-1} \cap \hat{S}| = 1$.
5. If $j = 2$, then either $Y_1 \cap \hat{S} = \emptyset$ or $|Y_1 \cap \hat{S}| = 1$.

Proof. Due to Lemma 4.8, there exists a GD-sequence S of G such that $\cup_{r=1}^i X_r \subseteq \hat{S}$ for some $i \in [k]$ and $\cup_{r=j}^k Y_r \subseteq \hat{S}$ for some $j \in [k]$, where $j \geq i$. It also satisfies the properties (1) and (2). Let S be such a sequence with the smallest index j . So, we have, $Y_{j-1} \cap \hat{S} \neq Y_{j-1}$. So, there is a vertex x of $N(Y_{j-1})$ which appears to footprint vertices of $\hat{S} \setminus Y_{j-1}$. Now, we need to show the properties (3), (4) and (5).

To prove (3), we assume that $j \geq i+3$. So, $Y_{i+2} \cap \hat{S} \neq Y_{i+2}$. This means that a vertex of Y_{i+2} is not appearing in S and so, a vertex of $N(Y_{i+2})$ appears in S , but $N(Y_{i+2}) = \cup_{r=i+2}^k X_r$ and we have, $(\cup_{r=i+2}^k X_r) \cap \hat{S} = \emptyset$. So, no vertex of $N(Y_{i+2})$ appears in S , a contradiction. Thus, property (3) holds.

To prove (4), let $j \geq 3$. Now, we show that $(\cup_{r=1}^{j-2} Y_r) \cap \hat{S} = \emptyset$. On the contrary, suppose that $Y_{t_0} \cap \hat{S} \neq \emptyset$ for some $t_0 \in \{1, \dots, j-2\}$. Let y be the vertex of $Y_{t_0} \cap \hat{S}$ which appears first in S . We consider two cases here.

Case 1: y footprints itself.

If y appears to footprint itself then the vertex x appears after y . Since x footprints some vertices of Y_{j-1} , we have, $x \in \{X_{j-1}, X_j, X_{j+1}\}$. To see this, if $x \in \cup_{r=j+2}^k X_r$, then $j \leq i - 1$, a contradiction on $j \geq i$. So, $x \in \{X_{j-1}, X_j, X_{j+1}\}$ holds. Now, we see that no vertex $x_0 \in X$ appears in S such that $N(x) \subset N(x_0)$. To see this, let such a vertex x_0 appears in S then either it appears before x or after x . Since x footprinted some vertices of Y_{j-1} , x_0 appears after x . Next, we get that x_0 appears for some vertex $y_0 \in \cup_{r=j}^k Y_r$ and so, y_0 is not appearing in S before x_0 . After the appearance of x_0 , all vertices of the set $N[y_0]$ are footprinted. So, $y_0 \notin \hat{S}$, a contradiction on $\cup_{r=j}^k Y_r \subseteq \hat{S}$. Thus, no vertex $x_0 \in X$ appears in S such that $N(x) \subset N(x_0)$. Hence, $x \in X_i$ or $x \in X_{i+1}$. If $x \in X_i$, $X_{i+1} \cap \hat{S} = \emptyset$. Now,

we modify the sequence by replacing the vertex x by vertices of $\hat{S} \setminus Y_{j-1}$ and get a new GD-sequence S' in which we have that $\cup_{r=1}^{i-1} X_r \subseteq \hat{S}'$ and $\cup_{r=j-1}^k Y_r \subseteq \hat{S}'$, where $j \geq i$. It also satisfies properties (1) and (2). This is a contradiction on our assumption that j is the smallest index satisfying all these mentioned properties. Therefore, y does not footprint itself.

Case 2: y does not footprint itself.

In this case, we get that all vertices of Y_{t_0} are footprinted by some vertex of X , which appears before y in S . So, y footprints some vertices from the set $\cup_{r=t_0}^k X_r$. Thus, $|Y_{t_0} \cap \hat{S}| = 1$ and $Y_{t_0} \cap \hat{S} = \{y\}$. Recall that x is the vertex which footprints vertices of $\hat{S} \setminus Y_{j-1}$ in S . Then, there can be two subcases:

Subcase 2.1: y appears before x .

In this subcase, we again get that $x \in X_i$ or $x \in X_{i+1}$. If $x \in X_i$, $X_{i+1} \cap \hat{S} = \emptyset$. We can prove it using the similar arguments as we gave in case 1. We modify S by replacing the vertex x by all vertices of $\hat{S} \setminus Y_{j-1}$. Thus, we get another GD-sequence S' in which we have that $\cup_{r=1}^{i-1} X_r \subseteq \hat{S}'$ and $\cup_{r=j-1}^k Y_r \subseteq \hat{S}'$, where $j \geq i$. It also satisfies properties (1) and (2). This is a contradiction on our assumption that j is the smallest index satisfying all these mentioned properties. Therefore, y does not appear before x .

Subcase 2.2: y appears after x .

Here y footprints some vertices of $\cup_{r=t_0}^k X_r$. Note that $t_0 \leq j - 2 < i + 1$. So, if y footprints a vertex $x_0 \in \cup_{r=t_0}^{j-2} X_r$ then x_0 appears nowhere in the sequence. This is a contradiction on the fact that $\cup_{r=1}^i X_r \subseteq \hat{S}$. So, y footprints a vertex of $\cup_{r=j-1}^k X_r$. Note that $Y_{j-1} \cap \hat{S} = \emptyset$ and $(\cup_{r=1}^{j-2} Y_r) \cap \hat{S} = \{y\}$. Now, we modify S by replacing the vertex y by the vertices of Y_{j-1} and get a new GD-sequence S' in which we have that $\cup_{r=1}^i X_r \subseteq \hat{S}'$ and $\cup_{r=j}^k Y_r \subseteq \hat{S}'$, where $j \geq i$. It also satisfies properties (1), (2), (3) and (4).

To prove (5), if $j = 2$ and $Y_1 \cap \hat{S} \neq \emptyset$ then either $|Y_1 \cap \hat{S}| = 1$ or $|Y_1 \cap \hat{S}| \geq 2$. In the latter case, we get that first vertex of Y_1 in S footprints itself. So, Observation 5 ensures that $|Y_1 \cap \hat{S}| = 1$.

Therefore, the lemma holds. □

Lemma 4.10. *Let S be a GD-sequence of G satisfying all properties of Lemma 4.9. Then the following statements are true:*

1. $j \in \{i, i+1\}$;
2. if $j = i$ then $|X_i| = 1$ or $|Y_i| = 1$;
3. if $j = i+1$, then either $|X_{i+1} \cap \hat{S}| = 1$ or $|Y_i \cap \hat{S}| = 1$.

Proof. Since S satisfies all properties of Lemma 4.9, there are integers $i \in [k-1], j \in \{2, \dots, k\}$ such that $\cup_{r=1}^i X_r \subseteq \hat{S}$ and $\cup_{r=j}^k Y_r \subseteq \hat{S}$. It also holds that either $X_{i+1} \cap \hat{S} = \emptyset$ or $|X_{i+1} \cap \hat{S}| = 1$. Similarly, either $Y_{j-1} \cap \hat{S} = \emptyset$ or $|Y_{j-1} \cap \hat{S}| = 1$. Using Lemma 4.9, we can also say that if $i \leq k-2$, then $(\cup_{r=i+2}^k X_r) \cap \hat{S} = \emptyset$ and, if $j \geq 3$, then $(\cup_{r=1}^{j-2} Y_r) \cap \hat{S} = \emptyset$. Lemma 4.9 ensures that $j \in \{i, i+1, i+2\}$.

To prove (1), we show that $j \neq i+2$. Suppose, $j = i+2$. Then we see that $\gamma_{gr}(G) \leq \alpha + \beta + 2$, where $\alpha = \sum_{r=1}^i |X_r|$ and $\beta = \sum_{r=i+2}^k |Y_r|$ due to Lemma 4.9. Now, consider the sequence $S_0 = (X_1) \oplus (X_2) \oplus \dots \oplus (X_i) \oplus (Y_k) \oplus (Y_{k-1}) \oplus \dots \oplus (Y_{i+2}) \oplus (y) \oplus (Y_{i+1})$, where $y \in Y_i$. If $|Y_{i+1}| > 1$, S_0 is a dominating sequence of G having length at least $\alpha + \beta + 3$, contradiction on S being a GD-sequence. This implies that $|Y_{i+1}| = 1$ and S_0 is also a GD-sequence of G which satisfies that $\cup_{r=i+1}^k Y_r \subseteq \hat{S}$ along with all properties of Lemma 4.9. So, $j = i+1$. Thus, property (2) holds.

To prove (2), we assume $j = i$. If X_i appears before Y_i , then an eventual second vertex from Y_i does not footprint any vertex, a contradiction. So, $|Y_i| = 1$. In the similar way, we get that $|X_i| = 1$, when Y_i appears before X_i . Thus, property (2) holds.

To prove (3), assume that $j = i+1$. We show that either $|X_{i+1} \cap \hat{S}| = 1$ or $|Y_i \cap \hat{S}| = 1$. So, first we assume that neither is true, that is, $|X_{i+1} \cap \hat{S}| = 0$ and $|Y_i \cap \hat{S}| = 0$. Now, if Y_{i+1} appears before X_i , then the length of S can be increased by including a vertex of X_{i+1} just before X_i , but S is a dominating sequence of G of maximum length. So, Y_{i+1} appears after X_i , thus length of S can be increased by including a vertex of Y_i just before Y_{i+1} , but S is a dominating sequence of G of maximum length. Hence, $|X_{i+1} \cap \hat{S}| = 1$ or $|Y_i \cap \hat{S}| = 1$. If $|X_{i+1} \cap \hat{S}| = 1$ and $|Y_i \cap \hat{S}| = 1$, then suppose that $X_{i+1} \cap \hat{S} = \{a\}$

and $Y_i \cap \hat{S} = \{b\}$. Clearly, vertices of the four sets $\{a\}$, $\{b\}$, X_i and Y_{i+1} appear in S . Let $\mathcal{K} = \{\{a\}, \{b\}, X_i, Y_{i+1}\}$. Recall that all vertices of X_i appear together in S . Similarly, all vertices of Y_{i+1} appear together in S . Let $A \in \mathcal{K}$ be the set whose vertices appear after the other three sets of \mathcal{K} in the sequence S . Then, all vertices of $N[A]$ are footprinted before the appearance of vertices of A . Therefore, either $|X_{i+1} \cap \hat{S}| = 1$ or $|Y_i \cap \hat{S}| = 1$. Thus property (3) holds. \square

Lemma 4.11. *Let S be a GD-sequence of G satisfying all properties of Lemma 4.9. Then the following statements are true for $i < k$ and $j \geq 2$:*

1. *If $j = i$ then, $X_{i+1} \cap \hat{S} = \emptyset$ and $Y_{i-1} \cap \hat{S} = \emptyset$;*
2. *If $j = i+1$, then either $|X_{i+1} \cap \hat{S}| = 1$ and $Y_i \cap \hat{S} = \emptyset$ or $|Y_i \cap \hat{S}| = 1$ and $X_{i+1} \cap \hat{S} = \emptyset$.*

Proof. To prove (1), let $j = i$ and $X_{i+1} \cap \hat{S} \neq \emptyset$. This implies that $|X_{i+1} \cap \hat{S}| = 1$. Let $X_{i+1} \cap \hat{S} = \{a\}$ and $\mathcal{K} = \{X_i, \{a\}, Y_i, Y_{i+1}\}$. Recall that all vertices coming from a set of open twins appear together in S . Let $A \in \mathcal{K}$ be the set whose vertices appear after the other three sets of \mathcal{K} in the sequence S . Then, all vertices of $N[A]$ are footprinted before the appearance of vertices of A , so A does not footprint a new vertex, a contradiction. Thus, $X_{i+1} \cap \hat{S} = \emptyset$. In the similar way we can prove that $Y_{i-1} \cap \hat{S} = \emptyset$.

To prove (2), we assume that $j = i+1$ and $|X_{i+1} \cap \hat{S}| = 1$. We need to show that $Y_i \cap \hat{S} = \emptyset$. On the contrary, suppose that $Y_i \cap \hat{S} \neq \emptyset$. Here, we see that S contain vertices from all of the sets X_i, X_{i+1}, Y_i and Y_{i+1} . Then there exists a vertex $a \in X_i \cup X_{i+1} \cup Y_i \cup Y_{i+1}$ whose closed neighborhood is footprinted before its appearance, a contradiction. In the similar way, we can prove that if $|Y_i \cap \hat{S}| = 1$, then $X_{i+1} \cap \hat{S} = \emptyset$. \square

Lemma 4.12. *Let S be a GD-sequence of G satisfying all properties of Lemma 4.7. If $i = k$, then one of the following statements is true.*

1. $\gamma_{gr}(G) = |X|$
2. $\gamma_{gr}(G) = |Y|$
3. $\gamma_{gr}(G) = |X| + |Y_k|$

Proof. If there exists a type (a) or type (b) GD-sequence of G then $\gamma_{gr}(G) = |Y|$ or $\gamma_{gr}(G) = |X|$. So, assume that all GD-sequences of G are of type (c). Now, consider the sequence S . It is given that $i = k$. Lemma 4.7 ensures that $j \geq i$. This further implies that $j = k$ and so, $Y_k \subseteq \hat{S}$. We get, $(\cup_{r=1}^{k-1} Y_r) = \emptyset$ due to Lemmas 4.9 and 4.11. Therefore, $|\hat{S}| \leq |X| + |Y_k|$ and $S_0 = (X_1) \oplus (X_2) \oplus \cdots \oplus (X_{k-2}) \oplus (Y_k) \oplus (X_k) \oplus (X_{k-1})$ is a dominating sequence of G . Therefore, $\gamma_{gr}(G) = |X| + |Y_k|$. \square

Analogous to Lemma 4.12, we give a symmetric lemma for the set Y of G , whose proof follows similar lines, and is omitted.

Lemma 4.13. *Let S be a GD-sequence of G satisfying all properties of Lemma 4.7. If $j = 1$, then one of the following statements is true.*

1. $\gamma_{gr}(G) = |Y|$
2. $\gamma_{gr}(G) = |X|$
3. $\gamma_{gr}(G) = |Y| + |X_1|$

Lemma 4.14. *If $G = (X, Y, E)$ is a chain graph such that every GD-sequence of G is of type (c), then for any GD-sequence S of G , the following statements are true:*

1. $\cup_{r=1}^i X_r \subseteq \hat{S}$ and $\cup_{r=j}^k Y_r \subseteq \hat{S}$ for some $i, j \in [k]$.

2. *Integers i and j satisfy exactly one of the following:*

(a) $i < k, j > 1$.

(b) $i = 1, j = 1$.

(c) $i = k, j = k$.

$$3. \text{ If } i < k, j > 1, \text{ then } \gamma_{gr}(G) = \begin{cases} \sum_{r=1}^i |X_r| + \sum_{r=j}^k |Y_r| & : j = i \\ \sum_{r=1}^i |X_r| + \sum_{r=j}^k |Y_r| + 1 & : j = i + 1 \end{cases}$$

4. *If $i = 1, j = 1$, then $\gamma_{gr}(G) = |Y| + |X_1|$*

5. *If $i = k, j = k$, then $\gamma_{gr}(G) = |X| + |Y_k|$*

Proof. Lemmas 4.6 and 4.7 ensure property (1). To prove property (2), we need to show that $i = k, j = 1$ cannot be true. So, assume that this is true. Then, Lemma 4.10 yields $k = 1$, a contradiction proving that property (2) holds. Property (3) follows from Lemmas 4.9. Properties (4) and (5) can be proved using Lemmas 4.12 and 4.13. \square

Now, We are ready to present an algorithm for computing a GD-sequence of a chain graph based on the above lemmas; see Algorithm 10. By following the above discussion, note that $\gamma_{gr}(G) \in A$, where

$$A = \left\{ n_1, n_2, n_1 + |Y_k|, n_2 + |X_1|, \sum_{r=1}^i |X_r| + \sum_{l=i}^k |Y_l|, \sum_{r=1}^i |X_r| + \sum_{l=i+1}^k |Y_l| + 1 \right\},$$

for some $i \in [k - 1]$. Thus, the sequence returned by Algorithm 10 is a GD-sequence of G . It is easy to see that Algorithm 10 computes S in linear time, which is the time needed to compute the parts $X_1, \dots, X_k, Y_1, \dots, Y_k$. The following theorem readily follows.

Theorem 4.15. *Algorithm 10 outputs a GD-sequence S of G in linear time.*

There is a connection between Grundy domination number of a graph and its independence number. Let A be an independent set of size $\alpha(G)$. By considering all vertices of A in any order, we get a closed neighborhood sequence of G , which yields the well known bound $\gamma_{gr}(G) \geq \alpha(G)$. In addition, for a chain graph G , we prove that Grundy domination number is either $\alpha(G)$ or $\alpha(G) + 1$.

Theorem 4.16. *If G is a chain graph, then $\gamma_{gr}(G) \in \{\alpha(G), \alpha(G) + 1\}$.*

Proof. Let G be a chain graph. By using the notation established in this section, we claim that $\alpha(G) \in \{n_1, n_2, \sum_{j=1}^i |X_j| + \sum_{j=i+1}^k |Y_j| \text{ for some } i \in [k - 1]\}$. To see this, let A be a maximum independent set of G . Three cases are possible. If $A \subseteq X$, then $A = X$ implying that $\alpha(G) = n_1$; if $A \subseteq Y$, then $A = Y$ implying that $\alpha(G) = n_2$. Now, if $A \cap X \neq \emptyset$ and $A \cap Y \neq \emptyset$, then one can easily infer that $A = (\cup_{j=1}^i X_j) \cup (\cup_{j=i+1}^k Y_j)$ for some $i \in [k - 1]$ implying that $\alpha(G) = \sum_{j=1}^i |X_j| + \sum_{j=i+1}^k |Y_j|$. Letting $i_0 \in [k - 1]$ be an index such that

Algorithm 10: GD-sequence of a chain graph

Input: A chain graph $G = (X, Y, E)$ without isolated vertices along with a chain ordering $(x_1, \dots, x_{n_1}, y_1, \dots, y_{n_2})$ of $V(G)$.

Output: A GD-sequence S of G .

```

1 Find the parts  $X_1, X_2, \dots, X_k$  and  $Y_1, Y_2, \dots, Y_k$ ;
2  $i = 0, \text{sum}[] = 0$ ;
3 if  $|Y_1| = 1$  then
4    $\text{sum}[0] = n_2 + |X_1|$ ;
5 else
6    $\text{sum}[0] = n_2$ ;
7 for  $i = 1 : k - 1$  do
8    $\text{sum}[i] = \sum_{j=1}^i |X_j| + n_2 - \sum_{j=1}^i |Y_j| + 1$ ;
9 if  $|X_k| = 1$  then
10   $\text{sum}[k] = n_1 + |Y_k|$ ;
11 else
12   $\text{sum}[k] = n_1$ ;
13 Find an index  $i^* \in \{0, 1, 2, \dots, k\}$  for which  $\text{sum}[i]$  is maximum;
14 if  $i^* == 0$  and  $|Y_1| > 1$  then
15    $S \leftarrow (Y_k) \oplus (Y_{k-1}) \oplus \dots \oplus (Y_1)$ ;
16 else if  $i^* == 0$  and  $|Y_1| = 1$  then
17   if  $k \geq 3$  then
18      $S \leftarrow (X_1) \oplus (Y_k) \oplus \dots \oplus (Y_3) \oplus (Y_1) \oplus (Y_2)$ ;
19   else
20      $S \leftarrow (X_1) \oplus (Y_1) \oplus (Y_2)$ ;
21 else if  $i^* == k$  and  $|X_k| > 1$  then
22    $S \leftarrow (X_1) \oplus (X_2) \oplus \dots \oplus (X_k)$ ;
23 else if  $i^* == k$  and  $|X_k| = 1$  then
24   if  $k \geq 3$  then
25      $S \leftarrow (X_1) \oplus (X_2) \oplus \dots \oplus (X_{k-2}) \oplus (Y_k) \oplus (X_k) \oplus (X_{k-1})$ ;
26   else
27      $S \leftarrow (Y_2) \oplus (X_2) \oplus (X_1)$ ;
28 else
29   choose a vertex  $x \in X_{i^*+1}$ 
30   if  $i^* > 1$  then
31      $S \leftarrow (X_1) \oplus (X_2) \oplus \dots \oplus (X_{i^*-1}) \oplus (Y_k) \oplus (Y_{k-1}) \oplus \dots \oplus (Y_{i^*+1}) \oplus x \oplus (X_{i^*})$ ;
32   else
33      $S \leftarrow (Y_k) \oplus (Y_{k-1}) \oplus \dots \oplus (Y_2) \oplus x \oplus (X_1)$ ;
34 return  $S$ .
```

$\sum_{j=1}^{i_0} |X_j| + \sum_{j=i_0+1}^k |Y_j| \geq \sum_{j=1}^i |X_j| + \sum_{j=i+1}^k |Y_j|$ for each $i \in [k-1]$, we may write $\alpha(G) \in \{n_1, n_2, \sum_{j=1}^{i_0} |X_j| + \sum_{j=i_0+1}^k |Y_j|\}$.

Algorithm 10 computes a GD-sequence of G and it turns out that $\gamma_{gr}(G) \in \{n_1, n_2, n_1 + |Y_k|, n_2 + |X_1|, \sum_{j=1}^i |X_j| + \sum_{j=i}^k |Y_j|, \sum_{j=1}^i |X_j| + \sum_{j=i+1}^k |Y_j| + 1\}$ for some $i \in [k-1]$. Note that $\gamma_{gr}(G) = n_1 + |Y_k|$ when $|X_k| = 1$ implying that $\gamma_{gr}(G) = \sum_{j=1}^{k-1} |X_j| + |Y_k| + 1$. Similarly, $\gamma_{gr}(G) = n_2 + |X_1|$ when $|Y_1| = 1$ implying that $\gamma_{gr}(G) = |X_1| + \sum_{j=2}^k |Y_j| + 1$. Now, if $\gamma_{gr}(G) = \sum_{j=1}^i |X_j| + \sum_{j=i}^k |Y_j|$ for some $i \in [k-1]$ then Lemma 4.10 ensures that $\gamma_{gr}(G)$ is either $\sum_{j=1}^i |X_j| + \sum_{j=i+1}^k |Y_j| + 1$ or $\sum_{j=1}^{i-1} |X_j| + \sum_{j=i}^k |Y_j| + 1$. Hence, $\gamma_{gr}(G) \in \{n_1, n_2, \sum_{j=1}^i |X_j| + \sum_{j=i+1}^k |Y_j| + 1 \text{ for some } i \in [k-1]\}$. Since Algorithm 10 computes the GD-sequence by finding the maximum of the set $\{n_1, n_2\} \cup \{\sum_{j=1}^i |X_j| + \sum_{j=i+1}^k |Y_j| + 1 : i \in [k-1]\}$, we have, $\gamma_{gr}(G) \in \{n_1, n_2, \sum_{j=1}^{i_0} |X_j| + \sum_{j=i_0+1}^k |Y_j| + 1\}$. Now, suppose $t = \sum_{j=1}^{i_0} |X_j| + \sum_{j=i_0+1}^k |Y_j|$, then we can write that $\alpha(G) \in \{n_1, n_2, t\}$ and $\gamma_{gr}(G) \in \{n_1, n_2, t+1\}$. Now, we consider three cases.

Case 1: $\gamma_{gr}(G) = n_1$:

In this case, $n_1 \geq n_2$ and $n_1 \geq t+1 > t$. This implies that $\alpha(G) = n_1$.

Case 2: $\gamma_{gr}(G) = n_2$:

In this case, $n_2 \geq n_1$ and $n_2 \geq t+1 > t$. This implies that $\alpha(G) = n_2$.

Case 3: $\gamma_{gr}(G) = t+1$:

In this case, if $\alpha(G) = n_1$ then $n_1 \geq t = \gamma_{gr}(G) - 1$. So, $\gamma_{gr}(G) \leq \alpha(G) + 1$. Similarly, if $\alpha(G) = n_2$ then $n_2 \geq t = \gamma_{gr}(G) - 1$. So, $\gamma_{gr}(G) \leq \alpha(G) + 1$. Otherwise, $\alpha(G) = t$ implying that $\gamma_{gr}(G) = \alpha(G) + 1$.

Therefore, $\gamma_{gr}(G) \in \{\alpha(G), \alpha(G) + 1\}$. □

Note that the GDD problem in the class of co-chain graphs (that is, the complements of chain graphs) is easily solvable. Indeed, in a co-chain graph $G = (X, Y, E)$, one can find similar partitions of X and Y into k sets X_1, \dots, X_k , and Y_1, \dots, Y_k , respectively, that arises from the \sim relation in G . Then, one can also immediately infer that $\gamma_{gr}(G) = k$.

4.3 Double Dominating Sequences

In this section, we first prove that the GD2D problem is NP-complete for split, bipartite and co-bipartite graphs. Next, we propose a linear-time algorithm that outputs a GDD-sequence for a threshold graph, a subclass of split graphs. We also present a linear-time algorithm for the GD2 problem in chain graphs, a subclass of bipartite graphs.

In the paper [44], Haynes and Hedetniemi proposed several variations of vertex sequences that could be interesting and gave a few initial results on some of these concepts. In particular, they presented Grundy double domination and found a formula for the Grundy double domination number in a square grid of an arbitrary dimension. They also proved that the Grundy double domination number of a tree T is exactly the number of vertices of T [44]. To the best of our knowledge, no hardness result is known for this variant in the literature.

The following lemma holds for any connected graph.

Lemma 4.17. *Let S be a GDD-sequence of a connected graph G . If $u, v \in V(G)$ such that $N[u] \subseteq N[v]$ and $u, v \in \widehat{S}$, then either u appears before v in the sequence S or there exists another GDD-sequence in which u appears before v .*

Proof. We may assume that u and v are not closed twins for otherwise the statement is clear. If v appears before u in the sequence S , then exchanging the places of these two vertices we obtain a sequence S' , and we claim that S' is also a double neighborhood sequence. Indeed, if u appears to dominate some vertex x , the second time, with respect to S , it dominates x the first time with respect to S' , and note that x is then dominated for the second time (since $x \in N[v]$). Hence when we consider v in S' , it dominates x for the second time. In addition, for all vertices w that lie between v and u in S , we note that if w was appearing to dominate some vertex $y \notin N[v] \setminus N[u]$, the first or the second time, with respect to S , then it does the same with respect to S' . Otherwise, if w was appearing to dominate some vertex $x \in N[v] \setminus N[u]$, then it dominates x the second time (x is dominated for the first time by v). However, in the sequence S' , w dominates x first time and v dominates x the second time. Since for all other vertices (that do not lie between v and u in S) nothing changes in S' , we

derive that S' is indeed a double neighborhood sequence of G , and since it is of the same length as S , it is a GDD-sequence of G . \square

4.3.1 NP-completeness Results

In this subsection, we prove that the GD2D problem is NP-complete for split, bipartite and co-bipartite graphs. The NP-completeness result for the split graph is the first hardness result for this problem which also implies the NP-hardness of the problem in general graphs.

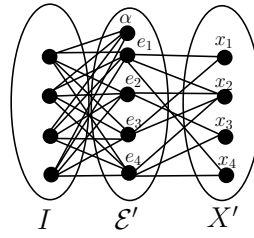
4.3.1.1 Split Graphs

A graph G is a *split graph* if $V(G)$ can be partitioned into two sets I and K , where I is an independent set and K is a clique. We may assume that a partition is done in such a way that $\alpha(G) = |I|$, which implies that every vertex in K has a neighbor in I . The partition $V(G) = [I, K]$ is a *split partition* of $V(G)$. The following observation holds for any split graph.

Lemma 4.18. *If $G = (I, K, E)$ is a split graph, then there exists a GDD-sequence S such that all vertices of I belong to \widehat{S} .*

Proof. Let G be a split graph and $[I, K]$ a split partition of G . Let S be a GDD-sequence of G , which maximizes the number of vertices from I . Suppose that there exists $x \in I$, which is not in S , and let $v_i \in \widehat{S}$ be the vertex that dominates x for the second time (noting that every vertex of G needs to be dominated twice). Since $x \notin \widehat{S}$, we infer that $v_i \in K$. Hence, $N[x] \subset N[v_i]$. Thus, the sequence S' , which is obtained from S by replacing v_i with x , is a double dominating sequence, since x dominates itself the second time and all other vertices of S' dominate the same vertices as in S . However, since $|\widehat{S'}| = |\widehat{S}|$, S' is a GDD-sequence, which yields a contradiction with the assumption that S maximizes the number of vertices from I . \square

Theorem 4.19. *The GD2D problem is NP-complete even when restricted to split graphs.*

FIGURE 4.4: Construction of bipartite graph G from the hypergraph H .

Proof. It is clear that the GD2D problem is in NP. To prove the NP-hardness, we reduce the GCD problem for hypergraphs to the GD2D problem for split graphs. Given a hypergraph $H = (X, \mathcal{E})$ with $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_m\}$, $|X| = n$ and $|\mathcal{E}| = m$, ($n, m \geq 2$), we construct an instance G of the GD2D problem, where G is a split graph, as follows.

Let $V(G) = I \cup K$, where $I = X$ and $K = \{e_1, e_2, \dots, e_m\}$. Vertices of I correspond to the vertices of X and vertices of K correspond to the hyperedges of H . For an edge $\mathcal{E}_i \in \mathcal{E}$, we denote the corresponding vertex in K by e_i . The edge set of G is given by $\{e_i e_j : 1 \leq i < j \leq m\} \cup \{ue_i : u \in \mathcal{E}_i, u \in I, i \in [m]\}$. Clearly, the set of vertices I is an independent set in G . The subgraph $G[K]$ is a complete graph. So, the graph G is a split graph with split partition $[I, K]$. See Fig. 4.4, which presents the construction of the graph G from a hypergraph H , which is given by $(X = \{x_1, x_2, x_3, x_4\}, \mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4, \mathcal{E}_5\})$, where $\mathcal{E}_1 = \{x_1, x_2, x_4\}$, $\mathcal{E}_2 = \{x_2, x_3\}$, $\mathcal{E}_3 = \{x_1, x_2\}$, $\mathcal{E}_4 = \{x_2, x_3, x_4\}$ and $\mathcal{E}_5 = \{x_1, x_3, x_4\}$.

Now, we prove the following claim.

Claim 5. For the graphs G and H discussed above, $\gamma_{gr}^{\times 2}(G) = |X| + \rho_{gr}(H)$.

Proof. First, let $\mathcal{C} = \{\mathcal{C}_{i_1}, \mathcal{C}_{i_2}, \dots, \mathcal{C}_{i_k}\}$ be a Grundy edge covering sequence of H . Suppose $X = \{x_1, x_2, \dots, x_n\}$. Then, the sequence $(x_1, x_2, \dots, x_n, e_{i_1}, e_{i_2}, \dots, e_{i_k})$ is a double dominating sequence for G . So, $\gamma_{gr}^{\times 2}(G) \geq n + k = |X| + \rho_{gr}(H)$.

For the other side, we assume that S is a GDD-sequence of G in which all vertices of I appear. Existence of such a sequence is ensured by Lemma 4.18. Thus, $|\hat{S}| = |I| + k = |X| + k = n + k$. Note that no two vertices of K are closed twins in G due to its construction from the hypergraph H . Among vertices from K , let u and v be the first that appear in S . Clearly, u appears for a vertex in I . Since u and v are not closed twins, then either v also

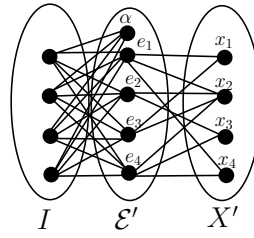
appears for a vertex in I , or $N[v] \subsetneq N[u]$, and all vertices in $N[v] \cap I$ appear in S before v . However, in the latter case we apply Lemma 4.17, and note that changing the places of u and v in S results in another GDD-sequence, in which both u and v dominate a vertex in I . Now, for all further vertices in S that are from K it is clear that they do not dominate a vertex in K (since u and v double dominate the entire K). Therefore, every vertex in S , which is from K , dominates a vertex in I either the first or the second time. So, S is a GDD-sequence in which all vertices coming from K dominate a vertex in I , all vertices of I appear in S and dominate themselves. Let S_K be the subsequence of vertices in S that are from K (in the same order as they appear in S). Clearly, S_K corresponds to the sequence of edges in H such that each edge of the sequence S_K covers a vertex in X that was not covered by edges appearing prior to it (since each vertex in S_K dominates a vertex in I). Thus S_K induces an edge covering sequence in H and so its length is at most $\rho_{gr}(H)$. This implies, $n + k \leq n + \rho_{gr}(H)$. So, $\gamma_{gr}^{\times 2}(G) \leq |X| + \rho_{gr}(H)$.

Hence, $\gamma_{gr}^{\times 2}(G) = |X| + \rho_{gr}(H)$ holds true. \square

Therefore, The GD2D problem is NP-complete even when restricted to split graphs. \square

4.3.1.2 Bipartite Graphs

Here, we prove the NP-completeness of the GD2D problem for bipartite graphs. We reduce the GCD problem for hypergraphs to the GD2D problem for bipartite graphs. Let $H = (X, \mathcal{E})$ be a hypergraph with no isolated vertices. It is known that the GCD problem is NP-hard in general graphs [20]. We claim that the GCD problem is efficiently solvable for $k \leq 2$. To see this, first, let $k = 1$. Now, the problem asks if there exists an edge covering sequence of a given hypergraph H with at least 1 hyperedge. To solve this problem, first we check whether union of all hyperedges of H equals vertex set of H or not. If yes, then the answer to the problem is YES otherwise the answer is NO. Thus, we have the solution of the GCD problem when $k = 1$, in polynomial-time. Now, let $k = 2$. Here, the problem asks if there exists an edge covering sequence of a given hypergraph H with at least 2 hyperedges. To solve this problem, first we check whether number of hyperedges of H is at least 2 or not. If not, clearly the answer to the problem is NO. So, we assume that H has at least two hyperedges. Next, we

FIGURE 4.5: Construction of bipartite graph G from the hypergraph H .

check whether union of all hyperedges of H equals vertex set of H or not. If not, the answer to the problem is NO. Otherwise, if the union of all hyperedges of H equals vertex set of H then, there are two possibilities: either there is a hyperedge E' which equals $V(H)$ or there is no such hyperedge. In the former case, there is one hyperedge E'' which is a proper subset of $V(H)$. Here, (E'', E') is a solution of size at least 2. So, the answer to the problem is YES. In the latter case, since union of all hyperedges of H equals vertex set of H , we get that any solution is of size at least 2. Thus, we have the solution of the GCD problem when $k = 2$, in polynomial-time. Therefore, the GCD problem is efficiently solvable for $k \leq 2$. Recall that the GCD problem is NP-complete for any $k \in \mathbb{Z}^+$. Consequently, the GCD problem is NP-complete for $k \geq 3$.

Theorem 4.20. *The GD2D problem is NP-complete for bipartite graphs.*

Proof. It is clear that the GD2D problem is in class NP. To show the NP-hardness, we give a polynomial reduction from the GCD problem in hypergraphs which is known to be NP-hard [20]. Given a hypergraph $H = (X, \mathcal{E})$ with $|X| = n$ and $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_m\}$, ($n, m \geq 2$), we construct an instance $G = (X^*, Y^*, E^*)$ of the GD2D problem, where G is a bipartite graph, as follows. $X^* = I \cup X'$ and $Y^* = \mathcal{E}'$, where $I = \{v_1, v_2, \dots, v_m\}$, $X' = \{x_1, x_2, \dots, x_n\}$ and $\mathcal{E}' = \{\alpha, e_1, e_2, \dots, e_m\}$. A vertex of X' corresponds to a vertex of X in the hypergraph H and the vertex e_i of \mathcal{E}' corresponds to the hyperedge \mathcal{E}_i of H . Now, a vertex x of X' is adjacent to a vertex of $e_i \in \mathcal{E}'$ in G if and only if $x \in \mathcal{E}_i$ in H . Each vertex of I is adjacent to each vertex of \mathcal{E}' in G . Clearly, G is a bipartite graph. Fig. 4.5 illustrates the construction of G when H is the hypergraph given by $(X = \{x_1, x_2, x_3, x_4\}, \mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4\})$, where $\mathcal{E}_1 = \{x_1, x_2, x_4\}$, $\mathcal{E}_2 = \{x_2, x_3\}$, $\mathcal{E}_3 = \{x_1, x_2\}$ and $\mathcal{E}_4 = \{x_2, x_3, x_4\}$.

Now, we show that $\rho_{gr}(H) \geq k$ if and only if $\gamma_{gr}^{\times 2}(G) \geq n + m + k + 1$, for $k \geq 3$. First, let $(\mathcal{E}_{i_1}, \mathcal{E}_{i_2}, \dots, \mathcal{E}_{i_{k'}})$ be an edge covering sequence of size at least k in H . Then the sequence $(x_1, x_2, \dots, x_n, v_1, v_2, \dots, v_m, \alpha, e_{i_1}, e_{i_2}, \dots, e_{i_{k'}})$ is a double dominating sequence of size at least $n + m + k + 1$ in G . So, we have $\gamma_{gr}^{\times 2}(G) \geq n + m + k + 1$.

For the converse part, we give a claim first.

Claim 6. There exists a double dominating sequence of G of size at least $n + m + k + 1$ in which the first vertex from \mathcal{E}' is the vertex α .

Proof. Let e_0 be the first vertex from \mathcal{E}' appearing in S . If $e_0 = \alpha$, then there is nothing to prove. So, we assume that $e_0 \neq \alpha$. Now, there can be two cases.

Case 1: $\alpha \notin \widehat{S}$

In this case, if e_0 is appearing to dominate vertices of I only, then we replace the vertex e_0 by the vertex α in the sequence S . Otherwise, e_0 is appearing to dominate some vertices from $X' \cup \{e_0\}$ also, then we put α just before e_0 in the sequence S . Hence, we modified the sequence S by removing at most 1 vertex and adding 1 new vertex. We see that S remains a double dominating sequence of size at least $n + m + k + 1$ in G and now, the first vertex from \mathcal{E}' appearing in S is α .

Case 2: $\alpha \in \widehat{S}$

Here, the vertex α appears in the sequence after e_0 . Since e_0 is the first vertex coming from \mathcal{E}' in S , it is appearing to dominate vertices of I and some vertices of X' . We have that $N[\alpha] = \{\alpha\} \cup I$. So, if we modify the sequence by putting the vertex α just before e_0 in the sequence, the updated sequence is also a double neighborhood sequence. This follows because e_0 dominates its neighbors from X' in the new sequence as they are not dominated by α . By doing this, size of S remains unchanged. So, S remains a double dominating sequence of size at least $n + m + k + 1$ in G and now, the first vertex from \mathcal{E}' appearing in S is α . Hence, the claim is true. \square

Let S be a double dominating sequence of size at least $n + m + k + 1$ in G satisfying Claim 6. Note that $|\widehat{S} \cap \mathcal{E}'| \geq k + 1$.

As $k \geq 3$, let e be the second vertex coming from \mathcal{E}' in S . Now, let A denotes the set of vertices appearing before the vertex α in S , B denotes the set of vertices appearing after the vertex α and before the vertex e . Finally, C denotes the set of vertices appearing after the vertex e in S .

Claim 7. $|I \cap (A \cup B)| \geq 2$.

Proof. Let $|I \cap (A \cup B)| \leq 1$, then we see that $|I \cap \widehat{S}| \leq 2$. Now, we get that $|\widehat{S}| = |\widehat{S} \cap I| + |\widehat{S} \cap \mathcal{E}'| + |\widehat{S} \cap X'| \leq 2 + m + 1 + n = n + m + 3 < n + m + 4$. This gives a contradiction as we considered S to be a double dominating sequence of size at least $n + m + k + 1$ in G , where $k \geq 3$. Hence, $|I \cap (A \cup B)| \geq 2$ holds true. \square

Claim 8. There exists a double dominating sequence S_0 of G of size at least $n + m + k + 1$ satisfying Claim 6 such that $\widehat{S}_0 \cap X' = X'$ and all vertices of X' appear before the vertex e in the sequence S_0 .

Proof. Here, we have two cases to consider.

Case 1: There is a vertex $x \in X'$ which does not appear in the sequence S :

Let $x \in X'$ be a vertex such that it does not appear in the sequence S . This tells that there are two vertices coming from \mathcal{E}' , say e_i and e_j which appear in S and they dominate the vertex x first and second time respectively. Let v^* denotes the vertex which appears just before the vertex e in the sequence S . Then, we see that the vertices e_i and e_j appear in the sequence S after v^* . Using Claim 7, we know that before the vertex e all vertices of \mathcal{E}' are dominated twice. So, vertex e_j is appearing to dominate vertices of X' only.

Now, there are two possibilities. First, assume that e_i is appearing only to dominate the vertex x first time, then we modify S by adding x just before e and removing the vertex e_i from S . But, if e_i was appearing to dominate some vertices of $\{e_i\} \cup I \cup (X' \setminus \{x\})$ also, then we modify S by putting the vertex x just before e in the sequence. By this modification, we removed at most 1 vertex from the sequence and added a new vertex to S . Thus, S remains a double dominating sequence of size at least $n + m + k + 1$ in G with x appearing in S before the vertex e .

Case 2: There is a vertex $x \in X' \cap \widehat{S}$ which appears after e in S :

In this case, vertex x is appearing to dominate itself only. Since all vertices of \mathcal{E}' are dominated twice before the vertex e , so we remove the vertex x from its place and put it just before e . Note that the size of S is not changed and so, S remains a double dominating sequence of size at least $n + m + k + 1$ in G with x appearing in S before the vertex e . Therefore, the claim holds true. \square

Claim 8 ensures that we can assume that $\widehat{S} \cap X' = X'$ and all vertices of X' appear before the vertex e in the sequence S . Combining all claims, we get that $|\widehat{S} \cap (\mathcal{E}' \setminus \{\alpha\})| \geq k$ and these vertices of $(\mathcal{E}' \setminus \{\alpha\})$ are appearing only to dominate vertices of X' second time. So, these vertices of $\widehat{S} \cap (\mathcal{E}' \setminus \{\alpha\})$ correspond to a legal hyperedge sequence of size at least k in the hypergraph H . So, $\rho_{gr}(H) \geq k$.

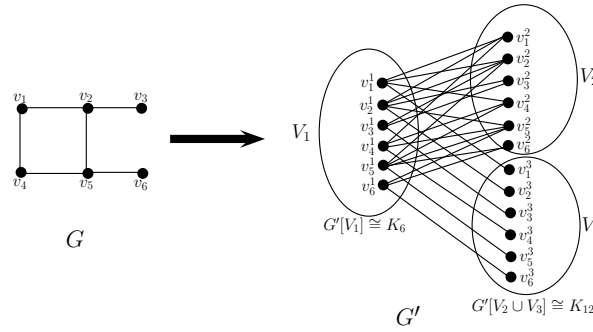
Therefore, the GD2D problem is NP-complete for bipartite graphs. \square

4.3.1.3 Co-bipartite Graphs

Here, we prove that the problem also remains NP-complete for co-bipartite graphs. For this, we give a polynomial reduction from the GDD problem in general graphs when $k \geq 4$. Given a graph $G = (V, E)$ with $V = \{v_1, v_2, \dots, v_n\}$ ($n \geq 2$), we construct an instance $G' = (V', E')$ of the GD2D problem in the following way.

Define the vertex set V' as $V' = V_1 \cup V_2 \cup V_3$, where $V_r = \{v_i^r : i \in [n]\}$ for each r , $1 \leq r \leq 3$. Add the edges in G' in the following way. (i) Add the edges so that $G'[V_1]$ and $G'[V_2 \cup V_3]$ are complete subgraphs of G' . (ii) If $v_j \in N_G[v_i]$, then add an edge between v_i^1 and v_j^2 . (iii) For each $i \in [n]$, add the edge $v_i^1 v_i^3$ in G' . Formally, define $E' = \{v_i^1 v_j^1, v_i^2 v_j^2, v_i^3 v_j^3 : 1 \leq i < j \leq n\} \cup \{v_i^2 v_j^3 : 1 \leq i \leq j \leq n\} \cup \{v_i^1 v_j^2 : v_j \in N_G[v_i]\} \cup \{v_i^1 v_i^3 : i \in [n]\}$. Clearly, G' is a co-bipartite graph. Fig. 4.6 illustrates the construction of G' from a graph G .

To prove the NP-hardness of the GD2D problem in co-bipartite graphs, it is enough to prove the following theorem.

FIGURE 4.6: Construction of co-bipartite graph G' from the graph G .

Theorem 4.21.* Let G' be the co-bipartite graph constructed from a graph $G = (V, E)$ with $V = \{v_1, v_2, \dots, v_n\}$ ($n \geq 2$) as explained above. Then, $\gamma_{gr}(G) \geq k$ if and only if $\gamma_{gr}^{\times 2}(G') \geq n + k$, for $k \geq 4$.

Proof. First, let $S = (v_{i_1}, v_{i_2}, \dots, v_{i_t})$ be a dominating sequence of G of size t , where $t \geq k$. Then the sequence $(v_1^3, v_2^3, \dots, v_n^3, v_{i_1}^2, v_{i_2}^2, \dots, v_{i_t}^2)$ is a double dominating sequence of G' of size at least $n + k$. So, we get that $\gamma_{gr}^{\times 2}(G') \geq n + k$.

Conversely, let S be a double dominating sequence of G' having size at least $n + k$. Now, we claim that there exists a double dominating sequence S^* of G' in which the following is true:

1. $V_1 \cap \widehat{S^*} = \emptyset$,
2. $V_3 \subseteq \widehat{S^*}$,
3. All vertices of V_3 appear at initial n places of the sequence S^* .

If S satisfies all the above conditions then there is nothing to prove. So, assume that $V_1 \cap \widehat{S} \neq \emptyset$. Then either $|V_1 \cap \widehat{S}| \geq 2$ or $|V_1 \cap \widehat{S}| \leq 1$.

If $|V_1 \cap \widehat{S}| \geq 2$, let v_i^1 and v_j^1 be the first two vertices of V_1 which are appearing in S . Let A be the subset of \widehat{S} which contains vertices of S appearing before v_i^1 in S . Let B be the subset of \widehat{S} which contains vertices of S appearing after v_i^1 in S and before v_j^1 . Finally, let C be the subset of \widehat{S} which contains vertices of S appearing after v_j^1 in S . Now, if $|(A \cup B) \cap (V_2 \cup V_3)| \leq 2$, we get that $|\widehat{S} \cap (V_2 \cup V_3)| \leq 2$ which further implies that $|\widehat{S}| \leq n + 2$. This contradicts the assumption that $k \geq 4$. So, we have that $|(A \cup B) \cap (V_2 \cup V_3)| \geq 3$.

In this case, we get that $C = \emptyset$. Since all vertices of $V_2 \cup V_3$ have been dominated twice before the appearance of v_j^1 and $v_i^1 \in V_1$ also appears before v_j^1 , we get that v_j^1 appears to dominate some vertex $v_k^1 \in V_1$ second time. As, no vertex of G' appears after v_j^1 , we have that $v_k^3 \notin \widehat{S}$. Now, we modify S by replacing the vertex v_j^1 by v_k^3 and get a new double dominating sequence of G' of same size. By repeating the above arguments, we can say that there exists a double dominating sequence of G' in which at most 1 vertex of V_1 appears. So, we assume that $|V_1 \cap \widehat{S}| \leq 1$.

First, we assume that $|V_1 \cap \widehat{S}| = 1$ and $V_1 \cap \widehat{S} = \{v_i^1\}$. Again, let A be the subset of \widehat{S} which contains vertices of S appearing before v_i^1 in S and B be the subset of \widehat{S} which contains vertices of S appearing after v_i^1 in S . Now, either $|A \cap (V_2 \cup V_3)| \leq 1$ or $|A \cap (V_2 \cup V_3)| \geq 2$. If $|A \cap (V_2 \cup V_3)| \leq 1$ then, $|\widehat{S}| \leq n + 3$. This contradicts the assumption that $k \geq 4$. So, we have that $|A \cap (V_2 \cup V_3)| \geq 2$. Note that v_i^1 appears after at least two vertices of $V_2 \cup V_3$ in S , this implies that v_i^1 appears only to dominate some vertex of V_1 first or second time. There can be two cases now.

Case 1: v_i^1 appears to dominate itself.

If there is a vertex $u \in V_2 \cup V_3$ which is a neighbor of v_i^1 in G' and $u \notin \widehat{S}$ then, we modify S by replacing the vertex v_i^1 by u and get a new double dominating sequence of G' of same size in which no vertex of V_1 appears. So, assume that all neighbors of v_i^1 belonging to the set $V_2 \cup V_3$ appear in \widehat{S} . So, $v_i^3 \in \widehat{S}$.

Thus, we have that v_i^3 appears to dominate v_i^1 first or second time and all other neighbors of v_i^1 from the set $V_2 \cup V_3$ belong to the set B . In particular, the vertex v_i^2 is also in B and it appears after v_i^3 in S . Note that v_i^2 appears to dominate some vertex v_j^1 of V_1 second time. This implies that $v_j^3 \notin \widehat{S}$. Now, we modify S by replacing the vertex v_i^2 by v_j^3 and the vertex v_i^1 by v_i^2 to get a new double dominating sequence of G' of same size in which no vertex of V_1 appears.

Case 2: v_i^1 is dominated twice before its appearance.

Here, suppose that v_i^1 appears to dominate some vertex v_j^1 of V_1 , first or second time. If $v_j^2 \notin \widehat{S}$, then we can modify S by replacing the vertex v_i^1 by v_j^2 to get a new double dominating sequence of G' of same size in which no vertex of V_1 appears. Similarly if $v_j^3 \notin \widehat{S}$, we get a

new double dominating sequence of G' of same size containing no vertex of V_1 . So, assume that $v_j^2, v_j^3 \in \widehat{S}$. Note that at least one of the vertices v_j^2 and v_j^3 does not belong to the set A . This implies that the vertex v_j^2 appears to dominate some vertex v_k^1 of V_1 second time and $v_k^3 \notin \widehat{S}$. Now, we modify S by replacing the vertex v_j^2 by v_k^3 and the vertex v_i^1 by v_j^2 to get a new double dominating sequence of G' of same size in which no vertex of V_1 appears.

Hence, we can assume that S contains no vertex of V_1 . Thus, condition (1) holds. Now, we need to show that $V_3 \subseteq \widehat{S}$. On the contrary, assume that this is not true. Let $v_i^3 \in V_3$ be a vertex which is not in \widehat{S} . This implies that the vertex v_i^1 is dominated both times by two vertices of V_2 . Let $v_j^2 \in V_2$ be the vertex which dominates v_i^1 second time. Now, we modify S by replacing the vertex v_j^2 by v_i^3 to get a new double dominating sequence of G' of same size in which the vertex v_i^3 appears. By repeating this argument, we get that there is a double dominating sequence of G' in which all vertices of V_3 appears. So, we can assume that $V_3 \subseteq \widehat{S}$ and thus, condition (2) is also satisfied.

Now, it remains to show that all n vertices of V_3 appear at initial n places. For this, it is enough to show that $v_i^3 \in V_3$ dominates the vertex v_i^1 first time for each $i \in [n]$. So, let v_i^1 be a vertex of V_1 such that it is dominated first time by a vertex v_j^2 of V_2 and second time by v_i^3 . Clearly v_i^3 appears after v_j^2 in S . Here, we see that $N_{G'}[v_i^3] \subseteq N_{G'}[v_j^2]$, so we can exchange the positions of these two vertices with each other and get a new double dominating sequence of G' of same size such that v_i^3 dominates v_i^1 first time. Hence, we get that all n vertices of V_3 appear at initial n places of S .

Therefore, we have that, at least k vertices of V_2 are appearing in S and all of them are appearing only to dominate vertices of V_1 second time. So, these vertices correspond to a dominating sequence of G of size at least k . Thus, $\gamma_{gr}(G) \geq k$. \square

4.3.2 Efficient Algorithms

In this subsection, we provide linear-time algorithms for two restricted graph classes. In the previous subsection, we showed that the GD2D problem is NP-complete for split graphs and bipartite graphs. To reduce the gap between the hierarchy of graph classes, we investigated the problem in threshold and chain graphs and obtained positive results.

Threshold graphs are a subclass of split graphs and chain graphs are a subclass of bipartite graphs.

4.3.2.1 Threshold Graphs

A *threshold graph* is a graph that can be constructed from the one-vertex graph by repeated applications of the following two operations:

1. Addition of a single isolated vertex to the graph.
2. Addition of a single dominating vertex to the graph, that is, a single vertex that is adjacent to all other vertices.

For notational convenience, we denote by I the set of vertices of G that were added to G by operation 1, while D denotes the set of vertices of G added by operation 2. The former vertices are called *isolated*, while the latter are *dominating vertices* of G (clearly, these should not be mistaken with the terms isolated and dominating in general context). We denote a threshold graph G by $G = (I, D, E)$.

Since double domination makes no sense in graphs with minimum degree 0, we may restrict our attention to the threshold graphs in which the construction ends with operation 2, which are exactly the connected threshold graphs. Note that threshold graphs are split graphs, where the set of isolated vertices forms an independent set I , while the set of dominating vertices forms a clique D .

Note that threshold graphs have a similar property as chain graphs, notably, the linear order in which vertices of I , resp. D , are processed when G is constructed, yields the inclusion-wise linear order of their open neighborhoods in D , resp. I . More formally, let x_1, \dots, x_k be the set of isolated vertices of a threshold graph G in the order in which they were added to G , and similarly, let y_1, \dots, y_ℓ be the set of dominating vertices of G in the order in which they were added to G (according to the above definition of threshold graphs). Then, $N(x_1) \supseteq N(x_2) \supseteq \dots \supseteq N(x_k)$ and $N(y_1) \cap I \subseteq N(y_2) \cap I \subseteq \dots \subseteq N(y_\ell) \cap I$. Given a graph G , once we obtain the parts D and I , an ordering of the vertices of I and D can be found by sorting the vertices according to their degrees in polynomial-time. Note that

the ordering obtained for the sets I and D via this method, will be the same order in which these vertices were added to the graph G .

Now, using Lemmas 4.18 and 4.17 we infer that the following can be assumed for a threshold graph G : there exists a GDD-sequence S of G such that all vertices of I belong to \widehat{S} and appear in the beginning of S . That is, $S = (x_k, \dots, x_1) \oplus S_D$, where $I = \{x_1, \dots, x_k\}$ is the set of isolated vertices of G , and S_D is a sequence, which consists of (some) vertices from D . It is also clear that the largest length of S_D , and thus of S , can be achieved if vertices of D are processed in the order in which they appear according to the definition of threshold graphs, where among closed twins (that is, dominating vertices between which lies no isolated vertex in the construction of G) only the first vertex is added to S_D .

The above yields the algorithm for obtaining a GDD-sequence S of a threshold graph G ; see Algorithm 11. In the input, the ordering (y_1, \dots, y_ℓ) of vertices of D in which they appear in G according to the definition of threshold graphs is given. As mentioned above, this implies $N(y_1) \cap I \subseteq N(y_2) \cap I \subseteq \dots \subseteq N(y_\ell) \cap I$.

Algorithm 11: GDD-sequence of a threshold graph

Input: A connected threshold graph $G = (I, D, E)$ along with an ordering (y_1, \dots, y_ℓ) of the vertices of D and an ordering (x_1, \dots, x_k) of the vertices of I .

Output: A GDD-sequence S of G .

```

1  $S = (x_k, x_{k-1}, \dots, x_1, y_1)$ ;
2  $i = 1$ ;
3 while  $i \leq \ell$  do
4   if  $N(y_i) \cap I = N(y_{i+1}) \cap I$  then
5      $i = i + 1$ ;
6   else
7      $S = S \oplus y_i$ ;
8      $i = i + 1$ ;
9 Output  $S$ .
```

Theorem 4.22. *Algorithm 11 returns a GDD-sequence of a threshold graph G . Provided that the ordering of dominating vertices is given (i.e., the ordering in which they are added to G according to the defining construction of G), the complexity of the algorithm is $O(n)$, where n is the order of G .*

Due to [45], threshold graphs can be recognized in linear time and the partition of its vertex set into D and I can also be computed in linear time. Thus, the preprocessing does not (significantly) worsen the computational complexity. Note that the complexity of Algorithm 11 is in fact $O(\ell)$, where ℓ is the number of dominating vertices of G , which can be better than $O(n)$.

4.3.2.2 Chain Graphs

Now, we present a linear-time algorithm to solve the GD2 problem in chain graphs. Let $G = (X, Y, E)$ denote a chain graph and P_X, P_Y be the twin partition of X, Y obtained by the relation \sim , respectively. Recall that, $P_X = \{X_1, X_2, \dots, X_k\}$ and $P_Y = \{Y_1, Y_2, \dots, Y_k\}$. Let $(x_1, x_2, \dots, x_{n_1}, y_1, y_2, \dots, y_{n_2})$ be the chain ordering of G .

For $i \in [k]$, x^i denotes the vertex of X_i having minimum index in the chain ordering of G . Similarly, y^i denotes the vertex of Y_i having maximum index in the chain ordering of G . Below, we give a result that gives the Grundy double domination number of a complete bipartite graph.

Proposition 13. Let $G = (X, Y, E)$ be a complete bipartite graph. Then $\gamma_{gr}^{\times 2}(G) = \max\{|X|, |Y|\} + 1$.

Proof. Without loss of generality, assume that $n_1 \geq n_2$. We show that $\gamma_{gr}^{\times 2}(G) = n_1 + 1$. Clearly $(x_1, x_2, \dots, x_{n_1}, y_1)$ is a double dominating sequence of G which implies that $\gamma_{gr}^{\times 2}(G) \geq n_1 + 1$.

Now, assume that S is a GDD-sequence of G such that $|\widehat{S}| > n_1 + 1$. This implies that S contains at least two vertices from both X and Y . Moreover, S contains at least two vertices from X side or Y side and exactly two vertices from the other side. Let v be the vertex which appears at the end of S .

First, we assume that $|\widehat{S} \cap X| = 2$. Note that $v \in X$ in this case. Since at least two vertices of Y side and one vertex of X side have been appeared before v in S , we have that, all vertices of X are dominated twice and all vertices of Y are dominated at least once before the appearance of v . Hence, we get that v appears to dominate some vertex $y \in Y$ second time

and $y \notin \widehat{S}$. Now, we modify S by replacing the vertex v by y and get a new GDD-sequence of G in which we have only one vertex from X side. So, $\gamma_{gr}^{\times 2}(G) \leq n_2 + 1 \leq n_1 + 1$ which contradicts our assumption that $|\widehat{S}| > n_1 + 1$. So, $|\widehat{S} \cap X| \neq 2$.

Now, suppose that $|\widehat{S} \cap Y| = 2$. Note that $v \in Y$ in this case and v appears to dominate some vertex $x \in X$ second time. Again, we have that $x \notin \widehat{S}$. Now, we modify S by replacing the vertex v by x and get a new GDD-sequence of G in which we have only one vertex from Y side. Again, $\gamma_{gr}^{\times 2}(G) \leq n_1 + 1$ which contradicts our assumption that $|\widehat{S}| > n_1 + 1$. So, $|\widehat{S} \cap Y| \neq 2$. Thus, we get that no such S exists. Therefore, $\gamma_{gr}^{\times 2}(G) = n_1 + 1$. \square

For technical reasons, we actually consider a slightly more generalized problem in chain graphs. Let $G = (X, Y, E)$ be a chain graph and $M \subseteq V(G)$. Vertices of M are called *marked vertices* of G . All remaining vertices of G are called *unmarked vertices*. We denote the set of unmarked vertices of G by V_0 and the subgraph of G induced on the set V_0 by G_0 . The set of marked vertices satisfy all the conditions written in equation 4.1.

$$M \subseteq (X_k \cup Y_1), |M \cap X_k| \leq 1, |M \cap Y_1| \leq 1, |X_k \setminus M| \geq 1, |Y_1 \setminus M| \geq 1 \quad (4.1)$$

A sequence $S = (v_1, v_2, \dots, v_k)$, where $v_i \in V_0$ for each $i \in [k]$, is called an *M-double neighborhood sequence* of (G, M) if for each i , the vertex v_i dominates at least one vertex u of G which is dominated at most once by its preceding vertices in the sequence S . In addition, if \widehat{S} is a double dominating set of G_0 , then we call S an *M-double dominating sequence* of (G, M) . Note that \widehat{S} may not be a double dominating set of G . An M-double dominating sequence with maximum length is called a *Grundy M-double dominating sequence* of (G, M) . The length of a Grundy M-double dominating sequence of (G, M) is called the *Grundy M-double domination number* of (G, M) and is denoted by $\gamma_{grm}^{\times 2}(G, M)$. Given a chain graph G and $M \subseteq V(G)$ satisfying equation 4.1, the **GRUNDY M-DOUBLE DOMINATION (GMD2)** problem asks to compute a Grundy M-double dominating sequence of (G, M) .

From this point, $\mathcal{G} = (G, M)$ denotes an instance of the GMD2 problem, where $G = (X, Y, E)$ is a chain graph and M is a subset of $V(G)$ satisfying equation 4.1. Let S be a Grundy M-double dominating sequence of \mathcal{G} . If $M = \emptyset$ then, S is also a GDD-sequence of G . So, the GD2 problem is a special case of the GMD2 problem.

Now, we state two important lemmas. The proofs of these lemmas are easy and, hence are omitted.

Lemma 4.23. *Let $M \neq \emptyset$. Then, $\gamma_{grm}^{\times 2}(\mathcal{G}) \leq \gamma_{gr}^{\times 2}(G)$.*

Lemma 4.24. *For any Grundy M -double dominating sequence S of \mathcal{G} , we have that $X_k \cap \widehat{S} \neq \emptyset$ and $Y_1 \cap \widehat{S} \neq \emptyset$.*

We prove a lemma for complete bipartite graphs that forms the basis of our algorithm.

Lemma 4.25. $\gamma_{grm}^{\times 2}(\mathcal{G}) \in \{\max\{n_1, n_2\}, \max\{n_1, n_2\} + 1\}$, when G is a complete bipartite graph.

Proof. There are four cases to consider.

Case 1: $M = \emptyset$: In this case, $\gamma_{grm}^{\times 2}(\mathcal{G}) = \gamma_{gr}^{\times 2}(G)$. Using Proposition 13, we have $\gamma_{grm}^{\times 2}(\mathcal{G}) = \max\{n_1, n_2\} + 1$.

Case 2: $M \cap X_k = \{x_{n_1}\}$ and $M \cap Y_1 = \emptyset$: Since $|M \cap X_k| = 1$, we have that $n_1 \geq 2$. We consider two subcases now.

Subcase 2.1: $n_1 = \max\{n_1, n_2\}$:

Here, we have that $\gamma_{gr}^{\times 2}(G) = n_1 + 1$. Now, if $n_2 = 1$, $\gamma_{grm}^{\times 2}(\mathcal{G}) \leq |X| - 1 + |Y| = n_1$. As the sequence $(x_1, x_2, \dots, x_{n_1-1}, y_1)$ is an M -double dominating sequence of \mathcal{G} of length n_1 . So, $\gamma_{grm}^{\times 2}(\mathcal{G}) = n_1 = \max\{n_1, n_2\}$. Otherwise, if $n_2 > 1$, the sequence $(x_1, x_2, \dots, x_{n_1-1}, y_1, y_2)$ is an M -double dominating sequence of \mathcal{G} of length $n_1 + 1$. Thus, we have that $\gamma_{grm}^{\times 2}(\mathcal{G}) = n_1 + 1 = \max\{n_1, n_2\} + 1$ using Lemma 4.23.

Subcase 2.2: $n_2 = \max\{n_1, n_2\}$:

Here, we have that $\gamma_{gr}^{\times 2}(G) = n_2 + 1$. Since $n_1 \geq 2$, we have that $n_2 \geq 2$. The sequence $(y_1, y_2, \dots, y_{n_2}, x_1)$ is an M -double dominating sequence of \mathcal{G} of length $n_2 + 1$. Thus, we have $\gamma_{grm}^{\times 2}(\mathcal{G}) = n_2 + 1 = \max\{n_1, n_2\} + 1$ using Lemma 4.23.

Case 3: $M \cap Y_1 = \{y_1\}$ and $M \cap X_k = \emptyset$:

This case is similar to case 2.

Case 4: $M \cap X_k = \{x_{n_1}\}$ and $M \cap Y_1 = \{y_1\}$:

Clearly, $n_1 \geq 2$ and $n_2 \geq 2$. We again consider two subcases.

Subcase 4.1: $n_1 = \max\{n_1, n_2\}$:

Here, we have that $\gamma_{gr}^{\times 2}(G) = n_1 + 1$. If $n_2 \geq 3$, the sequence $(x_1, x_2, \dots, x_{n_1-1}, y_2, y_3)$ is an M-double dominating sequence of \mathcal{G} of length $n_1 - 1 + 2 = n_1 + 1$. So, $\gamma_{grm}^{\times 2}(\mathcal{G}) = n_1 + 1 = \max\{n_1, n_2\} + 1$ using Lemma 4.23. But, if $n_2 = 2$, the sequence $(x_1, x_2, \dots, x_{n_1-1}, y_2)$ is an M-double dominating sequence of \mathcal{G} of length $n_1 - 1 + 1 = n_1$. So, $\gamma_{grm}^{\times 2}(\mathcal{G}) = n_1 = \max\{n_1, n_2\}$ using the fact that $\gamma_{grm}^{\times 2}(G) \leq |X| - 1 + |Y| - 1 = n_1 - 1 + 2 - 1 = n_1$.

Subcase 4.2: $n_2 = \max\{n_1, n_2\}$:

Similar to the subcase 4.1, we can prove that $\gamma_{grm}^{\times 2}(\mathcal{G})$ is either n_2 or $n_2 + 1$. \square

Algorithm 12 computes a Grundy M-double dominating sequence of \mathcal{G} based on the Lemma 4.25, when G is a complete bipartite graph. Next, we state some lemmas for \mathcal{G} when G is not a complete bipartite graph, that is, $k \geq 2$.

Lemma 4.26. *If there exists a Grundy M-double dominating sequence S^* of \mathcal{G} such that $|X_k \cap \widehat{S^*}| \geq 3$, then exactly one of the following is true:*

1. $\gamma_{grm}^{\times 2}(\mathcal{G}) = |X| + k$.
2. $\gamma_{grm}^{\times 2}(\mathcal{G}) = |X| + k - 1$.

Proof. Let $X_k = \{a_{k_1}, a_{k_2}, \dots, a_{k_t}\}$. There can be two cases.

Case 1: $M \cap X_k = \emptyset$

In this case, no vertex of X_k is marked and $(x_1, x_2, \dots, x_{n_1}, y^k, y^{k-1}, \dots, y^1)$ is an M-double dominating sequence of \mathcal{G} which implies that $\gamma_{grm}^{\times 2}(\mathcal{G}) \geq |X| + k$.

Since, $|X_k \cap \widehat{S^*}| \geq 3$, we get that $t \geq 3$. Note that we can assume that all vertices of $X_k \cap \widehat{S^*}$ appear in the same order as in the chain ordering. We also assume that all vertices of $(X_k \setminus \{a_{k_1}\}) \cap \widehat{S^*}$ appear together in S^* . Now, we show that there exists an M-double dominating sequence of \mathcal{G} in which all vertices of X_k appear. To see this, suppose that there is a vertex a_{k_i} , where $i \geq 4$, such that $a_{k_i} \notin \widehat{S^*}$. This means that there are two vertices

Algorithm 12: $S = \text{GrundyM1}(G, M)$ **Input:** $\mathcal{G} = (G, M)$, where $G = (X, Y, E)$ is a complete bipartite graph and $M \subseteq V(G)$ satisfying equation 4.1, $X = \{x_1, \dots, x_{n_1}\}$ and $Y = \{y_1, \dots, y_{n_2}\}$.**Output:** A Grundy M-double dominating sequence S of \mathcal{G} .

```

if  $M = \emptyset$  then
    if  $n_1 \geq n_2$  then
         $S = (x_1, x_2, \dots, x_{n_1}, y_1)$ 
    else
         $S = (y_1, y_2, \dots, y_{n_2}, x_1)$ 
if  $M \cap X_k = \{x_{n_1}\}$  and  $M \cap Y_1 = \emptyset$  then
    if  $n_1 \geq n_2$  then
        if  $n_2 = 1$  then
             $S = (x_1, x_2, \dots, x_{n_1-1}, y_1)$ 
        else
             $S = (x_1, x_2, \dots, x_{n_1-1}, y_1, y_2)$ 
    else
         $S = (y_1, y_2, \dots, y_{n_2}, x_1)$ 
if  $M \cap Y_1 = \{y_1\}$  and  $M \cap X_k = \emptyset$  then
    if  $n_2 \geq n_1$  then
        if  $n_1 = 1$  then
             $S = (y_2, y_3, \dots, y_{n_2}, x_1)$ 
        else
             $S = (y_2, y_3, \dots, y_{n_2}, x_1, x_2)$ 
    else
         $S = (x_1, x_2, \dots, x_{n_2}, y_2)$ 
if  $M \cap X_k = \{x_{n_1}\}$  and  $M \cap Y_1 = \{y_1\}$  then
    if  $n_1 \geq n_2$  then
        if  $n_2 \geq 3$  then
             $S = (x_1, x_2, \dots, x_{n_1-1}, y_2, y_3)$ 
        else
             $S = (x_1, x_2, \dots, x_{n_1-1}, y_2)$ 
    else
        if  $n_1 \geq 3$  then
             $S = (y_2, y_3, \dots, y_{n_2}, x_1, x_2)$ 
        else
             $S = (y_2, y_3, \dots, y_{n_2}, x_1)$ 
return  $S$ .

```

$y, y' \in Y \cap \widehat{S^*}$ which dominate a_{k_i} first and second time respectively. Note that y' appears after a_{k_3} in S^* . Here, we modify the sequence by replacing y' with the vertex a_{k_i} and get a new Grundy M-double dominating sequence of \mathcal{G} in which the vertex a_{k_i} appears. By repeating this argument, we get a Grundy M-double dominating sequence of \mathcal{G} in which

all vertices of X_k appear. So, assume that $X_k \subseteq \widehat{S}^*$. We also assume that all vertices of $X_k \setminus \{a_{k_1}\}$ appear together in S^* .

Next, we show that there exists a Grundy M-double dominating sequence of \mathcal{G} in which all vertices of X appear. So, let x_0 be a vertex of X side which does not appear in S^* . This implies that there is a vertex $y_0 \in Y$ which appears in S^* , to dominate x_0 the second time. Note that y_0 appears after all vertices of X_k in S^* . We modify S^* by replacing y_0 with the vertex x_0 and get a new Grundy M-double dominating sequence of \mathcal{G} in which the vertex x_0 appears. By repeating this argument, we get a Grundy M-double dominating sequence S of \mathcal{G} in which all vertices of X appear. Thus, $X \subseteq \widehat{S}$. Since $N(X_k) = Y$ and $|X_k \cap \widehat{S}| = |X_k| \geq 3$, we have that at most one vertex of Y appears before a_{k_3} in S . So, at most k vertices can appear in S from the Y side. Thus, $\gamma_{grm}^{\times 2}(\mathcal{G}) \leq |X| + k$ which further implies that $\gamma_{grm}^{\times 2}(\mathcal{G}) = |X| + k$.

Case 2: $M \cap X_k \neq \emptyset$

Let a_{k_t} be the marked vertex of X_k . Here, we see that $t \geq 4$. If $(X \setminus \{a_{k_t}\}) \not\subseteq \widehat{S}^*$, we can do similar modifications as done in case 1 and get a new Grundy M-double dominating sequence S of \mathcal{G} such that all vertices of $X \setminus \{a_{k_t}\}$ appear in S . Again, we see that at most k vertices can appear from the Y side in S . Thus, $\gamma_{grm}^{\times 2}(\mathcal{G}) \leq |X| - 1 + k$ which further implies that $\gamma_{grm}^{\times 2}(\mathcal{G}) = |X| + k - 1$. \square

Similar to Lemma 4.26, we state another lemma for the Y side of G . Proof of Lemma 4.27 is similar to the Lemma 4.26.

Lemma 4.27. *If there exists a Grundy M-double dominating sequence S^* of \mathcal{G} such that $|Y_1 \cap \widehat{S}^*| \geq 3$, then exactly one of the following is true:*

1. $\gamma_{grm}^{\times 2}(\mathcal{G}) = |Y| + k$.
2. $\gamma_{grm}^{\times 2}(\mathcal{G}) = |Y| + k - 1$.

Lemma 4.28. *Let \mathcal{G} be an instance of the GMD2 problem such that there is no Grundy M-double dominating sequence S^* of \mathcal{G} satisfying $|X_k \cap \widehat{S}^*| \geq 3$ or $|Y_1 \cap \widehat{S}^*| \geq 3$. Assume that S is a Grundy M-double dominating sequence of \mathcal{G} such that $|X_k \cap \widehat{S}| = 2$. Then either*

$|Y_1 \cap \widehat{S}| = 1$ or there exists another Grundy M-double dominating sequence S' of \mathcal{G} satisfying one of the following:

(1) $|X_k \cap \widehat{S}'| = 2$ and $|Y_1 \cap \widehat{S}'| = 1$. (2) $|X_k \cap \widehat{S}'| = 1$ and $|Y_1 \cap \widehat{S}'| = 2$.

Proof. Since \mathcal{G} has no Grundy M-double dominating sequence having at least 3 vertices from Y_1 , we have that $|Y_1 \cap \widehat{S}| \leq 2$. Moreover, Lemma 4.24 ensures that $|Y_1 \cap \widehat{S}| = 1$ or $|Y_1 \cap \widehat{S}| = 2$. Now, if $|Y_1 \cap \widehat{S}| = 1$, there is nothing to prove. So, assume that $|Y_1 \cap \widehat{S}| = 2$. Suppose that $X_k \cap \widehat{S} = \{a, b\}$ and $Y_1 \cap \widehat{S} = \{c, d\}$. Note that the sequence S ends with a vertex of the set $\{a, b, c, d\}$. There are two cases to consider.

Case 1: S ends with c or d :

First, we assume that S ends with the vertex d . As all vertices of the set $\{a, b, c\}$ have appeared before d , we get that d appears to dominate some vertex x^* of X second time. Note that $x^* \notin \widehat{S}$. Now, if x^* is an unmarked vertex, we modify S by replacing the last vertex d by the vertex x^* and get a new Grundy M-double dominating sequence S' of \mathcal{G} such that $|Y_1 \cap \widehat{S}'| = 1$. Otherwise, x^* is a marked vertex of G . This means that $x^* \in X_k$. As d is dominating x^* second time and $N(X_k) = Y$, we get that $|Y \cap \widehat{S}| = 2$. In particular, $Y \cap \widehat{S} = \{c, d\} = Y_1 \cap \widehat{S}$. Since $k \geq 2$, there is a vertex $y_0 \in Y_2$ which is appearing nowhere in S . So, we modify S by replacing the last vertex d by the vertex y_0 and get a new Grundy M-double dominating sequence S' of \mathcal{G} such that $|Y_1 \cap \widehat{S}'| = 1$. Hence, \widehat{S}' is the desired Grundy M-double dominating sequence of \mathcal{G} . Similar arguments can be given when S ends with the vertex c .

Case 2: S ends with a or b :

In this case, we get a new Grundy M-double dominating sequence S' of \mathcal{G} such that $|X_k \cap \widehat{S}'| = 1$ and $|Y_1 \cap \widehat{S}'| = 2$ by doing similar modifications as done in case 1.

Therefore, the lemma holds. □

Lemma 4.29. *Let \mathcal{G} be an instance of the GMD2 problem such that there is no Grundy M-double dominating sequence S^* of \mathcal{G} satisfying $|X_k \cap \widehat{S}^*| \geq 3$ or $|Y_1 \cap \widehat{S}^*| \geq 3$. Assume*

that S is a Grundy M -double dominating sequence of \mathcal{G} such that $|X_k \cap \widehat{S}| = 1$. Then $Y_k \subseteq \widehat{S}$.

Proof. Since every vertex of Y_k has to be dominated at least twice by the vertices of S , $N(Y_k) = X_k$ and $|X_k \cap \widehat{S}| = 1$, we get, $Y_k \subseteq \widehat{S}$. \square

Similar to Lemma 4.29, we state another lemma for G .

Lemma 4.30. *Let \mathcal{G} be an instance of the GMD2 problem such that there is no Grundy M -double dominating sequence S^* of \mathcal{G} satisfying $|X_k \cap \widehat{S}^*| \geq 3$ or $|Y_1 \cap \widehat{S}^*| \geq 3$. Assume that S is a Grundy M -double dominating sequence of \mathcal{G} such that $|Y_1 \cap \widehat{S}| = 1$. Then $X_1 \subseteq \widehat{S}$.*

Using Lemmas 4.28, 4.29 and 4.30, we can directly state the following result.

Lemma 4.31. *Let \mathcal{G} be an instance of the GMD2 problem such that there is no Grundy M -double dominating sequence S^* of \mathcal{G} satisfying $|X_k \cap \widehat{S}^*| \geq 3$ or $|Y_1 \cap \widehat{S}^*| \geq 3$. Then one of the following is true:*

(1) *There exists a Grundy M -double dominating sequence S of \mathcal{G} such that $|X_k \cap \widehat{S}| = 1$ and $Y_k \subseteq \widehat{S}$.*

(2) *There exists a Grundy M -double dominating sequence S of \mathcal{G} such that $|Y_1 \cap \widehat{S}| = 1$ and $X_1 \subseteq \widehat{S}$.*

Let \mathcal{G} be an instance of the GMD2 problem such that there is no Grundy M -double dominating sequence S^* of \mathcal{G} satisfying $|X_k \cap \widehat{S}^*| \geq 3$ or $|Y_1 \cap \widehat{S}^*| \geq 3$. We call a Grundy M -double dominating sequence S of \mathcal{G} as a *type 1 optimal sequence* of \mathcal{G} if it satisfies that $|X_k \cap \widehat{S}| = 1$ and $Y_k \subseteq \widehat{S}$. Similarly, We call a Grundy M -double dominating sequence S of \mathcal{G} , a *type 2 optimal sequence* of \mathcal{G} if it satisfies that $|Y_1 \cap \widehat{S}| = 1$ and $X_1 \subseteq \widehat{S}$.

Lemma 4.32. *Let \mathcal{G} be an instance of the GMD2 problem. Then one of the following is true:*

(1) *There exists a type 1 optimal sequence of \mathcal{G} .*

(2) *There exists a type 2 optimal sequence of \mathcal{G} .*

Proof. If a vertex of X is marked, we assume that it is the vertex x_{n_1} and if a vertex of Y is marked, we assume that it is the vertex y_1 . If \mathcal{G} is an instance such that there is no Grundy M-double dominating sequence S^* of \mathcal{G} satisfying $|X_k \cap \widehat{S^*}| \geq 3$ or $|Y_1 \cap \widehat{S^*}| \geq 3$ then the statement is true using Lemma 4.31.

So, assume that there is a Grundy M-double dominating sequence S^* of \mathcal{G} such that $|X_k \cap \widehat{S^*}| \geq 3$ or $|Y_1 \cap \widehat{S^*}| \geq 3$. If $|X_k \cap \widehat{S^*}| \geq 3$ then, using Lemma 4.26, we get that $\gamma_{grm}^{\times 2}(\mathcal{G})$ is $|X| + k$ or $|X| + k - 1$. If $\gamma_{grm}^{\times 2}(\mathcal{G})$ is $|X| + k$, then $(x_1, x_2, \dots, x_{n_1}, y^k, y^{k-1}, \dots, y^1)$ is a type 2 optimal sequence of \mathcal{G} . Note that x_{n_1} is not a marked vertex of G in this case. Otherwise, if $\gamma_{grm}^{\times 2}(\mathcal{G})$ is $|X| + k - 1$, then $(x_1, x_2, \dots, x_{n_1-1}, y^k, y^{k-1}, \dots, y^1)$ is a type 2 optimal sequence of \mathcal{G} . Thus, if $|X_k \cap \widehat{S^*}| \geq 3$, there exists a type 2 optimal sequence of \mathcal{G} .

Similarly, if $|Y_1 \cap \widehat{S^*}| \geq 3$, then we get that there exists a type 1 optimal sequence of \mathcal{G} . This is ensured due to Lemma 4.27. \square

Finally, we state the lemma which completely characterizes the structure of an optimal solution for an instance of the GMD2 problem. The proof is easy and hence, is omitted.

Lemma 4.33. *Let \mathcal{G} be an instance of the GMD2 problem. Then one of the following is true:*

- (1) *There exists a type 1 optimal sequence S of \mathcal{G} in which the vertex of $X_k \cap \widehat{S}$ appear in the last.*
- (2) *There exists a type 2 optimal sequence S of \mathcal{G} in which the vertex of $Y_1 \cap \widehat{S}$ appear in the last.*

We use a dynamic programming approach to solve the GMD2 problem for an instance \mathcal{G} in Algorithm 13. Through Lemma 4.33, we characterized the structure of an optimal solution. Next, we define the optimal solution of the problem recursively in terms of the optimal solutions to subproblems. For GMD2 problem, we pick the subproblems as the problem of finding a Grundy M-double dominating sequence of $\mathcal{G}' = (G', M')$, where G' is a subgraph of G and $M' \subseteq V(G')$ satisfying equation 4.1.

Let S be a Grundy M-double dominating sequence of \mathcal{G} which is a type 1 optimal sequence of \mathcal{G} and the vertex of $X_k \cap \widehat{S}$ appear in the last. We also assume that all vertices

of Y_k appear together just before the vertex of X_k . Let G_1 denotes the subgraph of G induced on the set of vertices $(X \setminus X_k) \cup \{x_{t+1}\} \cup (Y \setminus Y_k)$, where $t = |X| - |X_k|$. Let $M_1 = \{x_{t+1}\} \cup (M \cap Y_1)$. Then the subsequence of S obtained by removing the last $|Y_k| + 1$ vertices of S is a Grundy M-double dominating sequence of (G_1, M_1) .

Similarly, if S is a type 2 optimal sequence of \mathcal{G} having the vertex of $Y_1 \cap \widehat{S}$ in the last and G_2 denotes the subgraph of G induced on the set of vertices $(Y \setminus Y_1) \cup \{y_t\} \cup (X \setminus X_1)$, where $t = |M \cap Y_1| + 1$. Again, assume that all vertices of X_1 appear together just before the vertex of Y_1 . Let $M_2 = \{y_t\} \cup (M \cap X_k)$. Then the subsequence of S obtained by removing the last $|X_1| + 1$ vertices of S is a Grundy M-double dominating sequence of (G_2, M_2) .

Now, we give the algorithm to compute a Grundy M-double dominating sequence of \mathcal{G} .

Algorithm 13: $S = \text{GrundyM}(G, M)$

Input: $\mathcal{G} = (G, M)$, where $G = (X, Y, E)$ is a chain graph and $M \subseteq V(G)$ satisfying equation 4.1. $X = \{x_1, \dots, x_{n_1}\}$ and $Y = \{y_1, \dots, y_{n_2}\}$.

Output: A Grundy M-double dominating sequence S of \mathcal{G} .

```

if  $k = 1$  then
     $S = \text{GrundyM1}(G, M)$ ;
    return  $S$ ;
else
     $t = |X| - |X_k|$ ,  $X'_{k-1} = X_{k-1} \cup \{x_{t+1}\}$ ;
    if  $k \geq 3$  then
         $X' = \cup_{i=1}^{k-2} X_i \cup X'_{k-1}$ ;
    else
         $X' = X'_{k-1}$ ;
     $G^1_{k-1} = G[X' \cup (Y \setminus Y_k)]$ ,  $M \cap X_k = \{x_{t+1}\}$ ;
     $S_1 = \text{GrundyM}(G^1_{k-1}, M) \oplus (Y_k) \oplus x_{t+1}$ ;
     $t = |M \cap Y_1| + 1$ ,  $Y'_1 = Y_2 \cup \{y_t\}$ ;
    if  $k \geq 3$  then
         $Y' = \cup_{i=3}^k Y_i \cup Y'_1$ ;
    else
         $Y' = Y'_1$ ;
     $G^2_{k-1} = G[(X \setminus X_1) \cup Y']$ ,  $M \cap Y_1 = \{y_t\}$ ;
     $S_2 = \text{GrundyM}(G^2_{k-1}, M) \oplus (X_1) \oplus y_t$ ;
    if  $|\widehat{S}_1| \geq |\widehat{S}_2|$  then
        return  $S_1$ ;
    else
        return  $S_2$ ;

```

Algorithm 13 computes a Grundy M-double dominating sequence of $\mathcal{G} = (G, M)$ by recursively appending some vertices at the end of the Grundy M-double dominating sequence of (G', M') , where G' is a subgraph of G . Note that this task can be performed in linear-time.

Based on the above discussion, we directly state the following theorem.

Theorem 4.34. *Algorithm 13 outputs a Grundy M-double dominating sequence of $\mathcal{G} = (G, M)$ in linear-time, where G is a chain graph.*

To solve the GD2 problem in a chain graph G , we compute a Grundy M-double dominating sequence of (G, \emptyset) using Algorithm 13. So, we can state the following theorem.

Theorem 4.35. *A GDD-sequence of a chain graph G can be computed in linear-time.*

4.4 Summary

We studied the GDD problem and the GD2D problem in this chapter. We proved that both problems are NP-complete for bipartite graphs and co-bipartite graphs. In addition, we showed that the GD2D problem is NP-complete for split graphs. On a positive note, we presented a linear-time algorithm to solve the GD problem in chain graphs. We also remarked on a connection between the independence number of a graph G and the Grundy domination number of G . We also proved that the GD2D problem is efficiently solvable for chain graphs and for threshold graphs. We solved this problem in chain graphs using a dynamic programming approach. Since the class of chain graphs is a subclass of bipartite graphs, the gap between the efficient algorithms and NP-completeness in the subclasses of bipartite graphs has been narrowed a little for both problems. As threshold graph is a subclass of split graphs, the same can be said for chordal graphs.

Chapter 5

Maximum Weighted Edge Biclique

5.1 Introduction

In this chapter, we discuss the MWEB problem for edge weighted bipartite graphs. In particular, we resolve the complexity status of the problem in chain graphs and bipartite permutation graphs when the weights of the edges of the graph are taken from the set of positive real numbers.

The MAXIMUM VERTEX BICLIQUE (MVB) problem is to find a biclique of a graph G with maximum number of vertices. The decision version of the MVB problem is NP-complete for general graphs [40], but the MVB problem is polynomial-time solvable for bipartite graphs [40]. The MAXIMUM EDGE BICLIQUE (MEB) problem is to find a biclique in an unweighted graph G with the maximum number of edges. The decision version of the MEB problem is NP-complete for general graphs [40] and it also remains NP-complete for bipartite graphs [79] unlike the MVB problem. Many researchers have also studied some other variations of these problems, see [40, 30, 29, 47]. The MEB problem was first introduced in [40] and further studied in [79, 68, 88, 73, 8]. Some wide details about the applications of the MEB problem can be found in [30, 73]. Since the MEB problem is hard to approximate in bipartite graphs within n^δ for some $\delta > 0$ [36, 42] under certain assumptions such as random 4-SAT or 3-SAT hardness hypothesis, researchers have also studied the problem for subclasses of bipartite graphs. The MEB problem is polynomial-time solvable for the following subclasses of bipartite graphs: chordal bipartite graphs, convex bipartite graphs, and bipartite permutation graphs [73, 5, 31, 32, 41, 51]. Some other hardness results are also available for the MEB problem based on some assumptions ([68, 8, 37, 7]). In this chapter, we study the MWEB problem which is the weighted version of the MEB problem. The WEBD problem is the decision version of the MWEB problem.

There exists a restricted version of the MWEB problem, namely the S -MWEB problem, where S is a subset of real numbers from which edge weights are taken and the input graph

is a bipartite graph. In 2008, Tan ([88]) proved that for a wide range of choices of S , no polynomial-time algorithm can approximate the S -MWEB problem within a factor of n^ϵ for some $\epsilon > 0$ unless $\text{RP}=\text{NP}$. He also proved that the decision version of the S -MWEB problem is NP-complete even for $S = \{-1, 0, 1\}$. In this work, we show that this problem remains NP-complete when $S = \{1, -M\}$ ($M > |E(G)|$) in a very restricted graph class. On the positive side, we show that when the set S is the set of positive real numbers, the S -MWEB problem is quadratic time solvable for bipartite permutation graphs and linear-time solvable for chain graphs.

The organization of upcoming sections of this chapter is as follows. In Section 5.2, we discuss the NP-completeness result for the WEBD problem. Section 5.3 presents the polynomial time algorithms for the problem when edge weights are taken from the set \mathbb{R}^+ .

5.2 NP-completeness Result

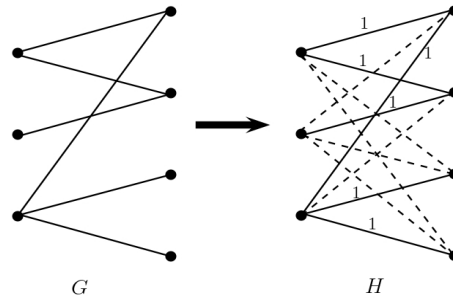
In this section, we show that the WEBD problem is NP-complete for complete bipartite graphs, a very restricted subclass.

Theorem 5.1. *The WEBD Problem is NP-complete for complete bipartite graphs.*

Proof. Clearly, the WEBD problem is in NP. To prove the NP-hardness of the WEBD problem for a complete bipartite graph, we make a polynomial reduction from the unweighted version of the same problem for bipartite graphs. So, we prove a construction of a weighted complete bipartite graph from an unweighted bipartite graph.

Let $G = (X, Y, E)$ be an unweighted bipartite graph with $|X| = n_1$ and $|Y| = n_2$. We construct a new graph H which is nothing but K_{n_1, n_2} . Now, for an edge e in H , we define its weight to be 1 if $e \in E$ and $-M$ otherwise, where $M > m = |E|$. So, H is a weighted complete bipartite graph with weights as any real number. Fig. 5.1 illustrates the construction of H from G . The dashed edges in Fig. 5.1 are the edges with weight $-M$.

Now to complete the proof of the theorem, we only need to prove the following claim.

FIGURE 5.1: An illustration to the construction of H from G .

Claim 5.2.1. G has a biclique of size at least $k > 0$ if and only if H has a biclique of weight at least $k > 0$.

Proof. Let C be a biclique of G of size at least $k > 0$. Clearly, C corresponds to a biclique of weight at least k in H . Conversely, let C be a biclique of weight at least k in H . If all the edges of C are of weight 1 then all the edges of C are present in G also and so, C corresponds to a biclique of size at least k in G . Now, if there is at least one edge e_0 of C having weight $-M$ then we see that $\sum_{e \in E(C) \setminus \{e_0\}} w(e) \geq k + M$. Since exactly $m = |E(G)|$ edges of H have weight 1, we have, $\sum_{e \in E(C) \setminus \{e_0\}} w(e) \leq m < M$, a contradiction to $\sum_{e \in E(C) \setminus \{e_0\}} w(e) \geq k + M$. So, no edge of C has weight $-M$. \square

Hence, the theorem is proved. \square

We have observed that the WEBD problem is NP-complete even for complete bipartite graphs. In the next section, we will discuss the S -MWEB problem with S as the set of positive real numbers, which will be the restricted version of the MWEB problem. Throughout the Section 5.3, by the MWEB problem we mean the S -MWEB problem, where $S = \mathbb{R}^+$.

5.3 Polynomial Time Algorithms

In this section, we design efficient algorithms for the \mathbb{R}^+ -MWEB problem in bipartite permutation graphs and chain graphs, subclasses of bipartite graphs. The class of chain graphs is a subclass of bipartite permutation graphs so the algorithm for bipartite permutation

graphs also works for chain graphs. We present separate algorithms for both of these graph classes for the sake of better running times.

5.3.1 Bipartite Permutation Graphs

Let $G = (X, Y, E)$ be a weighted bipartite permutation graph with the strong ordering $(<_X, <_Y)$ of G . Weights on the edges are some positive real numbers. Suppose that $<_X = (x_1, x_2, \dots, x_k)$ and $<_Y = (y_1, y_2, \dots, y_{k'})$. We write $u <_X v$ or $u <_Y v$ for vertices u, v of G if u appears before v in the strong ordering of vertices of G . We write $u < v$ when it is clear from the context that u, v are coming from which side of the bipartition. For any edge $x_i y_j$, its weight is denoted by w_{ij} .

Now, we define the first and last neighbor of a vertex in G . Since both $<_X$ and $<_Y$ satisfy adjacency property (see lemma 1.4), for a vertex v of G , its neighbor set has some consecutive vertices in $<_X$ or $<_Y$. *First neighbor* of v is defined as the vertex that appears first in the strong ordering of G in its neighbor set and *last neighbor* of v is defined as the vertex that appears last in the strong ordering of G in its neighbor set. For any vertex u of G , $f(u)$ denotes the first neighbor of u , and $l(u)$ denotes the last neighbor of u . For u in X , we denote $f(u)$ by y_{α_u} and $l(u)$ by y_{β_u} where $1 \leq \alpha_u \leq \beta_u \leq k'$.

It can be observed that for a bipartite permutation graph G with its strong ordering $(<_X, <_Y)$, it has the following properties which will be used in the further discussion (See [55]):

1. Given any vertex of G , its neighbor set consists of some consecutive vertices in $<_X$ or $<_Y$.
2. For a pair of vertices u, v from X or Y , if $u < v$ then $f(u) \leq f(v)$ and $l(u) \leq l(v)$.

Now, we discuss the structure of a maximal biclique of G which will be used in getting a maximum biclique of G .

Let $G' = (X', Y', E')$ denotes a maximal biclique of G with $X' = \{x_i, x_{i+1}, \dots, x_j\}$ and $Y' = \{y_{i'}, y_{i'+1}, \dots, y_{j'}\}$ then edge $x_i y_{i'}$ is called the *first edge* of G' . We call an edge uv of

G as a *safe edge* if it is the first edge of some maximal biclique of G . We will see that one safe edge corresponds to exactly one maximal biclique of G and vice versa.

Lemma 5.2. *Let $G' = (X', Y', E')$ be a biclique of G with $X' = \{x_i, x_{i+1}, \dots, x_j\}$ and $Y' = \{y_{i'}, y_{i'+1}, \dots, y_{j'}\}$, then G' is a maximal biclique of G if and only if the following holds for the graph G .*

$$(a) \ l(x_i) = y_{j'}$$

$$(b) \ f(x_j) = y_{i'}$$

$$(c) \ l(y_{i'}) = x_j$$

$$(d) \ f(y_{j'}) = x_i$$

Proof. First, let us assume that G' is a maximal biclique. We need to show that conditions (a), (b), (c) and (d) are true. For (a), it is clear that $l(x_i) \geq y_{j'}$ since G' is a biclique. If equality holds, we are done. So, let $l(x_i) > y_{j'}$, say $l(x_i) = y_t (> y_{j'})$. Since vertices are ordered according to the strong ordering, all vertices of X' are adjacent to the vertices $y_{j'+1}, y_{j'+2}, \dots, y_t$ in G implying that G' is not a maximal biclique of G . Now for (b), suppose that $f(x_j) < y_{i'}$, say $f(x_j) = y_p (< y_{i'})$. All vertices of X' are adjacent to $y_p, y_{p+1}, \dots, y_{i'-1}$ in G because of the strong ordering of the vertices of G , but G' was maximal. Similarly (c) and (d) can be proven.

Conversely, we assume that the conditions (a), (b), (c) and (d) are true. Let, if possible, G' is not maximal. Then there exists a vertex v in G for which one of the following conditions must be satisfied: (i) $v < x_i$ and $vy_{j'} \in E(G)$, (ii) $v < y_{i'}$ and $vx_j \in E(G)$, (iii) $v > x_j$ and $vy_{i'} \in E(G)$, and (iv) $v > y_{j'}$ and $vx_i \in E(G)$. But none of the edges $vy_{j'}, vx_j, vy_{i'}, vx_i$ can be present in G because of our assumption that (a), (b), (c) and (d) are true. So, G' is a maximal biclique. \square

For any edge $e = uv$, the biclique corresponding to e , is the subgraph induced by the vertices $\{u, \dots, l(v), v, \dots, l(u)\}$. By Lemma 5.2, it can be observed that any maximal biclique of G can be identified from its first edge(safe edge). Given any edge uv ($u \in X$ and $v \in Y$) of G , one can easily check whether that is a safe edge or not as follows: If the first neighbor of

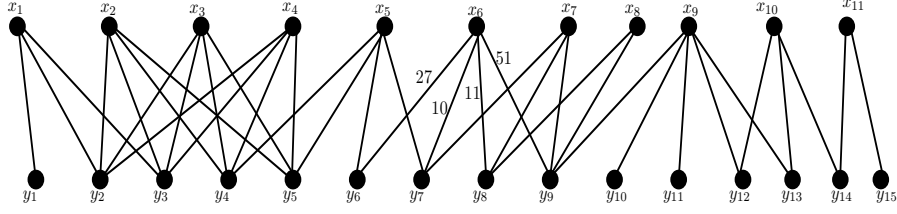


FIGURE 5.2: An example of Bipartite Permutation Graph.

the last neighbor of v is equal to v and first neighbor of last neighbor of u is equal to u , then uv qualifies as a safe edge. We observe from lemma 5.2 that this condition is both necessary and sufficient for a biclique (corresponding to an edge uv) to be a maximal biclique. Hence, we can say that number of safe edges in G is equal to the number of maximal bicliques of G . We denote the maximal biclique corresponding to the safe edge e by G_e . For every vertex u of G , we define an array called *prefix sum array (psa)* of u of size $d(u)$ as an array in which each value equals the sum of weights of edges up to that position starting from $f(u)$. The *psa* of x_i (or y_j) is denoted by $A_i[\]$ (or $B_j[\]$). Fig. 5.2 represents a bipartite permutation graph. Next, we illustrate all the terminologies defined in this section using Fig. 5.2.

In bipartite permutation graph shown in Fig. 5.2, x_2y_2 is a safe edge since $f(l(x_2)) = f(y_5) = x_2$ and $f(l(y_2)) = f(x_4) = y_2$ but x_4y_4 is not as $f(l(x_4)) = f(y_5) = x_2 \neq x_4$. Prefix sum array of the vertex x_6 is $A_6 = \{27, 37, 48, 99\}$, where $A_6[1] = 27$, $A_6[2] = 27 + 10 = 37$, $A_6[3] = 27 + 10 + 11 = 48$ and $A_6[4] = 27 + 10 + 11 + 51 = 99$.

Our Algorithm

Our idea for finding a maximum biclique is to look at all possible maximal bicliques of G and then return the one with the maximum weight. Since weights are positive real numbers any maximum biclique is some maximal biclique of G . The idea behind our algorithm is the following.

1. Find all safe edges of G .
2. Find *psa* of each vertex of G .
3. For each vertex $u \in X$,

- (a) for each $v \in N(u)$ (choose vertex v in the given ordering)
- (b) if $e = uv$ is a safe edge
 - i. find the maximal biclique G_e .
 - ii. find W_e , the weight of biclique G_e using psa of vertices.
- 4. Output the maximal biclique G_{e^*} for which W_{e^*} is maximum.

Note: We implement step 3 for each vertex in $O(n)$ -time and hence overall complexity of step 3 is $O(n^2)$. The detailed algorithm is given in Algorithm 14.

Theorem 5.3. *Algorithm 14 outputs a maximum weighted edge biclique of the bipartite permutation graph G .*

Proof. We know that all edge weights of G are positive and a MWEB of G is a maximal biclique of G . Hence, it is enough to show that Algorithm 14 finds all maximal bicliques of G and compares their weights. Suppose that H is a maximal biclique of G , then its first edge, say uv , is a safe edge and it will be identified while checking the condition $f(l(v)) == v$ and $f(l(u)) == u$. Hence, biclique H will be determined and its weight will be calculated during the execution of the algorithm. This proves that our algorithm computes the weights of all maximal bicliques of G , and hence it produces a maximum weighted edge biclique of G . \square

Theorem 5.4. *Algorithm 14 runs in $O(n^2)$ -time.*

Proof. For any edge $e \in E$, it will take constant time to check whether e qualifies as a safe edge or not. This is ensured due to Lemma 5.2. So, the preprocessing of all the safe edges takes $O(m)$ -time as it scans all the edges one by one. For a vertex u of G , calculating its psa will take $d(u)$ amount of time. Hence, finding psa of each vertex will take $O(m)$ -time. For a vertex $u \in X$, step 3 can be implemented in $O(n)$ -time. This is possible because, for all the safe edges in which one of the end point is u , we can find the weights of the corresponding bicliques in $O(n)$ -time altogether. So, overall step 3 takes

Algorithm 14: Algorithm for finding a MWEB of a bipartite permutation graph G

Input: A bipartite permutation graph $G = (X, Y, E)$ with the strong ordering of its vertices.

Output: A maximum weighted edge biclique of G .

```

/* identifying safe edges */
for each edge  $e = uv$  in  $E$  do
    if  $f(l(v)) == v$  and  $f(l(u)) == u$  then
        mark  $e$  as a safe edge
/* finding prefix sum arrays of vertices of  $G$  */
/* If  $N(x_i) = \{y_{s_1}, y_{s_2}, \dots, y_{s_{d(x_i)}}\}$ ,  $S_{x_i}$  denotes the set  $\{1, 2, \dots, d(x_i)\}$  */
for each vertex  $x_i$  from  $X$  do
    for every  $j$  from  $S_{x_i}$  do
         $A_i[j] = A_i[j-1] + w_{is_j}$ 
/* similarly we can find psa of vertices of  $Y$  */
max:=0
for each vertex  $x = x_i$  from  $X$  do
    sum:=0
    /*  $S'_x$  denotes the set  $\{y_{a_1}, y_{a_2}, \dots, y_{a_t}\}$  where  $a_1 < a_2 < \dots < a_t$ 
       such that  $xy_{a_1}, xy_{a_2}, \dots, xy_{a_t}$  are safe edges */
    for  $j = 1$  to  $t$  do
        /* finding maximal biclique corresponding to the safe
           edge  $e = xy_{a_j}$  */
         $X_e := \{x_i, x_{i+1}, \dots, x_p\}$  //  $x_p = l(y_{a_j})$ 
         $Y_e := \{y_{a_j}, y_{a_j+1}, \dots, y_q\}$  //  $y_q = l(x_i)$ 
         $E_e := \{uv | u \in X_e, v \in Y_e\}$ ,  $G_e := (X_e, Y_e, E_e)$ 
        if sum==0 then
            for each vertex  $x' = x_b$  from  $X_e$  do
                sum := sum +  $A_b[q - \alpha_{x'} + 1] - A_b[a_j - \alpha_{x'}]$ 
             $W_e := sum$ 
        else
             $W_e := sum + W_1 - W_2$ , sum :=  $W_e$ 
            /*  $W_1$  and  $W_2$  are the weights of the subgraphs
               induced by the vertices  $\{x_{c+1}, \dots, l(y_{a_j}), y_{a_j}, \dots, y_q\}$  and
                $\{x_i, \dots, x_c, y_{a_{j-1}}, \dots, y_{a_j-1}\}$  respectively, where
                $x_c = l(y_{a_{j-1}})$ .  $W_1$  and  $W_2$  are obtained using psa of
               vertices */
        if  $W_e > max$  then
            max :=  $W_e$ ,  $e^* := e$ 
return  $G_{e^*}$  and max

```

$O(n^2)$ -time. Therefore, the algorithm returns a maximum weighted edge biclique of G in $O(m) + O(m) + O(n^2) \approx O(n^2)$ -time. \square

5.3.2 Chain Graphs

Let $G = (X, Y, E)$ be a chain graph with the chain ordering $(x_1, x_2, \dots, x_{n_1}, y_1, y_2, \dots, y_{n_2})$ such that $N(x_1) \subseteq N(x_2) \subseteq \dots \subseteq N(x_{n_1})$ and $N(y_1) \supseteq N(y_2) \supseteq \dots \supseteq N(y_{n_2})$. Throughout this section, $G = (X, Y, E)$ denotes a weighted chain graph with $|X| = n_1$ and $|Y| = n_2$.

Recall the relation \sim on $X \cup Y$ discussed in Chapter 1. Let P_X be the twin partition of X and P_Y be the twin partition of Y . We write $P_X = \{X_1, X_2, \dots, X_k\}$ and $P_Y = \{Y_1, Y_2, \dots, Y_k\}$. Note that $[u]$ denotes the equivalence class for $u \in V(G)$.

Now, we define the representative vertex for each set of P_X . For a set $S \in P_X$, a vertex from S is called the *representative vertex* of the set S , if it is the least indexed vertex among all vertices of S . We denote the representative vertex of a set S by r_S . Next, we state some observations related to maximal bicliques of a chain graph which leads to a maximum weighted edge biclique of G .

Lemma 5.5. *Let $G' = (X', Y', E')$ be a maximal biclique of G , then the following holds:*

- (a) *If $x \in X'$, then $[x] \subseteq X'$.*
- (b) *If $y \in Y'$, then $[y] \subseteq Y'$.*

Proof. (a) Here, we will show that $[x] \subseteq X'$ for any $x \in X'$. Let $x_0 \in [x]$, as x_0 and x are similar vertices, $N(x_0) = N(x)$. Now, $Y' \subseteq N(x) = N(x_0)$ implies that x_0 is adjacent to all vertices of Y' in G . We must have these edges in G' as it is a maximal biclique. So, $[x] \subseteq X'$ is true.

Proof of part (b) is similar. \square

Below, we give a result that describes the detailed structure of a maximal biclique of a chain graph.

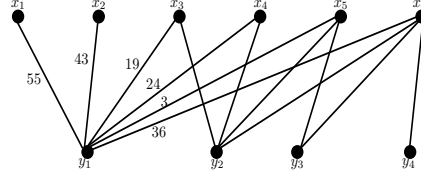


FIGURE 5.3: An example of Chain Graph.

Lemma 5.6. *Let $G' = (X', Y', E')$ be a maximal biclique of G . Then there exists an index $1 \leq i \leq k$ such that $X' = X_i \cup X_{i+1} \cup \dots \cup X_k$ and $Y' = N(r_{X_i})$.*

Proof. We know that vertices of G have an ordering as $\{x_1, x_2, \dots, x_{n_1}\}$ and $\{y_1, y_2, \dots, y_{n_2}\}$ for X and Y respectively. Let j be the minimum index from $\{1, 2, \dots, n_1\}$ such that $x_j \in X'$ and there is some t such that $x_j \in X_t$. Now Lemma 5.5 tells that $[x_j] = X_t \subseteq X'$ implying that $x_j = r_{X_t}$. Since j is the smallest index, we get that $\{X_1 \cup X_2 \cup \dots \cup X_{t-1}\} \cap X' = \phi$. Now, as $Y' \subseteq N(x_j)$ and G is a chain graph, $X' = X_t \cup X_{t+1} \cup \dots \cup X_k$. Hence, for $i = t$, one part of the lemma holds. For the remaining part, it is enough to show that $N(x_j) \subseteq Y'$. So, let y be a neighbor of x_j , then y is adjacent to all vertices in the set $\{x_{j+1}, x_{j+2}, \dots, x_{n_1}\}$ implying that $y \in Y'$. Hence, $Y' = N(r_{X_i})$ and $X' = X_i \cup X_{i+1} \cup \dots \cup X_k$. \square

It can be identified from Lemma 5.6 that a chain graph has exactly k maximal bicliques, where k is the number of distinct equivalence classes corresponding to the relation \sim . Now, we define an array called *partition sum array(pts)* of size k for each $y \in Y$. In a partition sum array of a vertex y , each value contains the sum of weights of the edges incident on the vertex y coming from one set of P_X . We denote the *pts* of y_i by $A_i[\]$. Fig. 5.3 represents a chain graph. We illustrate all the terminologies defined in this section using Fig. 5.3.

In the chain graph shown in Fig. 5.3, the partition $P_X = \{X_1, X_2, X_3, X_4\}$, where $X_1 = \{x_1, x_2\}$, $X_2 = \{x_3, x_4\}$, $X_3 = \{x_5\}$ and $X_4 = \{x_6\}$. Partition sum array of the vertex y_1 is $A_1 = \{98, 43, 3, 36\}$, where $A_1[1] = 55 + 43 = 98$, $A_1[2] = 19 + 24 = 43$, $A_1[3] = 3$ and $A_1[4] = 36$.

By Lemma 5.6, we know the structure of maximal bicliques of G . One can easily see that each maximal biclique can be identified from the representative vertex of one of the X_i 's from P_X . We use the notation G_x for the maximal biclique corresponding to the

representative vertex x and, W_x for the weight of the maximal biclique G_x , where x is the representative vertex of some set in P_X .

Our Algorithm

Our basic idea for finding a maximum biclique in chain graphs is to find weight of each maximal biclique of G and output the one with the maximum weight. Since G has only k maximal bicliques, so, in order to get the desired biclique, we need to find out the weights of these k bicliques. Since chain graph is a subclass of bipartite permutation graph, we may also use Algorithm 14 to compute a maximum weighted edge biclique of G . The ordering of vertices of G as given in chain graph will also work for bipartite permutation graph. In this way, we will get our desired output in $O(n^2)$ -time. Here, we propose an algorithm in which we use a different method to find out the sum of each maximal biclique of G which results in overall running time $O(m + n)$. The difference here is to use partition sum array instead of prefix sum array. Below, we present the algorithm.

Algorithm 15: Algorithm for finding a MWEB of a chain graph G

Input: A chain graph $G = (X, Y, E)$ with the chain ordering of its vertices.

Output: A maximum weighted edge biclique of G .

1. Find the partitions $P_X = \{X_1, X_2, \dots, X_k\}$ and $P_Y = \{Y_1, Y_2, \dots, Y_k\}$ from the equivalence relation \sim , say $R = \langle r_{X_k}, r_{X_{k-1}}, \dots, r_{X_1} \rangle$.
 2. Calculate the ptsa for each vertex of Y .
 3. For each vertex u according to the order in which it appears in R ,
 find the maximal biclique G_u corresponding to the vertex u .
 find W_u , the weight of biclique G_u using ptsa of vertices from $Y \cap V(G_u)$.
 4. Output the maximal biclique G_{u^*} for which W_{u^*} is maximum.
-

Note that we implement step 3 for each vertex $u \in R$ such that $W_{r_{X_j}}$ is calculated using ptsa of vertices of $N(r_{X_j})$.

Proof of the correctness of Algorithm 15 follows from the fact that it considers weights of all maximal bicliques of G and any maximum biclique is one of the maximal bicliques. So, we can directly state the following theorem.

Theorem 5.7. *Algorithm 15 outputs a maximum weighted edge biclique of a chain graph G .*

To analyze the running time of Algorithm 15, we need to bring some notations into consideration. We denote the cardinalities of sets in the partition P_X and P_Y by p_i, q_j for X_i, Y_j respectively, i.e. $|X_i| = p_i$ and $|Y_j| = q_j$. Now, we give a result that will be used in analyzing the running time of Algorithm 15.

Lemma 5.8. *Let G be a chain graph with a partition obtained from the \sim relation defined on X as well as on Y . Then $m \geq kq_1 + (k-1)q_2 + \dots + q_k$.*

Proof. We know that $N(X_1) \subset N(X_2) \subset \dots \subset N(X_k)$ and $N(Y_1) \supset N(Y_2) \supset \dots \supset N(Y_k)$. Since $Y_1 \cup Y_2 \cup \dots \cup Y_k = Y$ and for $i \neq j$, $Y_i \cap Y_j = \emptyset$, we can write that

$$\begin{aligned} m &= \sum_{y \in Y_1} d(y) + \sum_{y \in Y_2} d(y) + \dots + \sum_{y \in Y_k} d(y) \\ &= q_1 \sum_{i=1}^k p_i + q_2 \sum_{i=2}^k p_i + \dots + q_k p_k \\ &\geq kq_1 + (k-1)q_2 + \dots + q_k. \end{aligned}$$

The last inequality follows since $|X_i| \geq 1$ for $1 \leq i \leq k$. □

Theorem 5.9. *Algorithm 15 runs in $O(m+n)$ -time.*

Proof. Step 1 will take $O(n)$ -time as we have to go through all the vertices of G . To find out the time taken by step 2, we see that we are doing some number of additions during Algorithm 2. For each vertex y of Y , we are doing $d(y)$ number of additions, so overall step 2 takes $\sum_{y \in Y} d(y) = O(m)$ time. Now, to analyze step 3, we see that in our proposed algorithm, we are finding weights of maximal bicliques in the order $W_{r_{X_k}}, W_{r_{X_{k-1}}}, \dots, W_{r_{X_1}}$. For calculating $W_{r_{X_j}}$, we are doing $\sum_{i=1}^j q_i + (j-1)$ number of additions, where j varies from k down to 1. Hence, step 3 performs

$$\sum_{i=1}^k q_i + \sum_{i=1}^{k-1} q_i + \dots + q_1 + (k-1) + (k-2) + \dots + 2 + 1 + 0 \leq kq_1 + (k-1)q_2 + \dots + q_k + \frac{k(k+1)}{2}$$

number of additions. Now we know that $m \geq \frac{k(k+1)}{2}$ since $N(X_1) \subset N(X_2) \subset \dots \subset N(X_{k_1})$ and $N(Y_1) \supset N(Y_2) \supset \dots \supset N(Y_{k_2})$. Now using Lemma 5.8, we can say that step 3 will take $O(m)$ -time to execute. Clearly, choosing the maximum among all the W_u 's will take $O(k)$ -time. Therefore, the Algorithm 15 returns a maximum weighted edge biclique of G in $O(n) + O(m) + O(m) + O(k) \approx O(m + n)$ time. \square

5.4 Summary

In this chapter, we discussed the MWEB problem. We proved that the decision version of the MWEB problem remains NP-complete even for complete bipartite graphs, which is a subclass of bipartite graphs. On the positive side, we show that for the input graph G , if the weight of each edge is a positive real number, then the MWEB problem is $O(n^2)$ -time solvable for bipartite permutation graphs and $O(m + n)$ -time solvable for chain graphs.

Chapter 6

Neighbor-Locating Coloring

6.1 Introduction

This chapter is dedicated to a variant of the VERTEX COLORING problem in graphs, namely “NEIGHBOR-LOCATING COLORING” (NLC) problem. We discuss certain bounds for general graphs as well as for some restricted graph classes such as chain graphs, proper interval graphs and co-bipartite graphs. We also present a linear-time approximation algorithm for the NLC problem. A comparison between the complexity of the NLC problem with the VERTEX COLORING problem is also discussed.

Graph coloring is a well-studied fundamental problem in graph theory, which involves assigning labels or “colors” to elements (such as vertices, edges, or both) of a graph while adhering to certain constraints. Although there are various forms of graph coloring, most research in this area has been focused on vertex coloring, which has gained significant attention since the famous four-color problem. The real-world applications of vertex coloring have attracted researchers from many engineering fields, including scheduling, timetabling, register allocation, frequency assignment, and many more.

Let $G = (V, E)$ be a simple and undirected graph. *Vertex coloring* of G is an assignment of colors to the vertices of G . When a vertex coloring of G uses k colors, we call it a k -coloring of G . A k -coloring of G can also be viewed as a function from V to $[k]$. A coloring of vertices of G is a *proper coloring* if no two adjacent vertices receive the same color. The VERTEX COLORING problem asks to find a proper coloring of G using the minimum number of colors. The minimum number of colors required for a proper coloring of G is called the *chromatic number* of G , denoted by $\chi(G)$. Let $c : V \rightarrow [k]$ be a proper coloring of G . Then the color assigned to a vertex $v \in V$ in the coloring c is denoted by $c(v)$. For a subset S of V , we denote the set of colors assigned to the vertices of S in the coloring c by $c(S)$. Malaguti et al. did a vast survey on vertex coloring in 2010 [67].

In literature, several variants of the VERTEX COLORING problem have been introduced. Recently, the notion of neighbor-locating colorings is introduced [3]. Particularly, we look at vertex colorings where any two vertices with the same color can be differentiated from one another by the colors of their respective neighbors. A proper coloring c of a graph G is called a *neighbor-locating coloring* (NL-coloring) if, for any two vertices u, v of the same color, $c(N(u)) \neq c(N(v))$. The *neighbor-locating chromatic number* of G is the minimum k such that a neighbor-locating k -coloring of G exists. We denote the neighbor-locating chromatic number of G by $\chi_{NL}(G)$. Given a graph G , the NEIGHBOR-LOCATING COLORING (NLC) problem requires assigning a color to each vertex of G such that the coloring is a neighbor-locating coloring and the number of colors used is $\chi_{NL}(G)$.

To the best of our knowledge, there is no hardness result on the NLC problem in the literature. Alcon et al. gave some bounds for the neighbor-locating chromatic number of a general graph [3]. In the same article, they examined the neighbor-locating chromatic number for some graph operations: the join and the disjoint union. Moreover, the neighbor-locating chromatic number of split graphs and mycielski graphs have been computed. Alcon et al. also established some bounds on the neighbor-locating chromatic number for unicyclic graphs and trees in another paper [4]. Alcon et al. gave neighbor-locating chromatic number of paths, cycles, fans, and wheels [2]. In 2020, Alcon et al. characterized all graphs having neighbor-locating chromatic number equal to n or $n - 1$, where n is the number of vertices in the graph [3]. In 2022, Mojdesht [69] studied the conjectures posed by Alcon et al. in [3].

The section-wise organization is as follows: Section 6.2 describes all the bounds we have obtained for the NLC problem in some special graph classes. Next, in Section 6.3, we give the approximation algorithm for the problem. Finally, section 6.4 remarks on a complexity difference between the VERTEX COLORING problem and the NLC problem.

6.2 Bounds

In this section, we discuss bounds for the neighbor-locating chromatic number of graphs in terms of different important graph parameters. We first provide some lower and upper bounds for general graphs. Then, we also discuss bounds for some special graph classes

such as chain graphs, proper interval graphs, and co-bipartite graphs. In addition, we provide exact values of the neighbor-locating chromatic number for restricted types of chain graphs.

First, we discuss some terminologies which will be used in this section. For a graph $G = (V, E)$, two vertices $u, v \in V$ are called open twins if $N(u) = N(v)$. A set S subset of V is called an *open twin set* of G if for any pair of vertices $u, v \in S$, u, v are open twins. We define a relation on the vertex set of graph G . Two vertices are related if and only if they are open twins. We call it the “open twin relation”. It is easy to verify that this relation is an equivalence relation on V . An equivalence class of the open twin relation is referred to as a *twin class*. The *twin number* of a graph G , denoted by $\tau(G)$, is the maximum cardinality of a twin class of G . Below, we state some bounds as propositions for the neighbor-locating chromatic number of a connected graph G .

Proposition 14. $\chi_{NL}(G) \geq \tau(G) + 1$.

Proof. Any neighbor-locating coloring of G requires that each pair of vertices from a twin class have unique colors assigned to both of its vertices. The size of a largest twin class is $\tau(G)$. Additionally, there exists a vertex outside the largest twin class that must be colored differently. Therefore, $\chi_{NL}(G) \geq \tau(G) + 1$ holds true. \square

Recall that $\beta(G)$ is the vertex cover number of G .

Proposition 15. $\chi_{NL}(G) \leq \beta(G) + \tau(G)$.

Proof. To prove this proposition, we give a NL-coloring c of G using $\beta(G) + \tau(G)$ colors. Let C be a minimum vertex cover of G . Assign each vertex of C , a unique color from the set $[\beta(G)]$. Next, partition the vertices of $V(G) \setminus C$ on the basis of open twin vertices, that is, if two vertices from $V(G) \setminus C$ are open twins, consider them in the same part. Let P denote this partition. Now, consider a set from the collection P and assign each vertex of that set a unique color from the set $\{\beta(G) + 1, \beta(G) + 2, \dots, \beta(G) + \tau(G)\}$. Repeat this for each set from the collection P . Note that we have assigned colors in such a way that this coloring is a proper coloring. When two vertices $u, v \in V(G) \setminus C$ have the same color, their neighborhoods in the set C are different as they are not open twins. Since each vertex of C

has a different color, $c(N(u)) \neq c(N(v))$. Thus, c is a NL-coloring of G using $\beta(G) + \tau(G)$ colors. Therefore, $\chi_{NL}(G) \leq \beta(G) + \tau(G)$. \square

When G contains no open twins, we have $\tau(G) = 1$. A graph is called *twin free* if no two vertices are open twins. Combining this with Proposition 15, Proposition 16 holds.

Proposition 16. For any twin-free graph G , $\chi_{NL}(G) \leq \beta(G) + 1$.

Note that for a bipartite graph $G = (X, Y, E)$, $\beta(G) \leq \min\{|X|, |Y|\}$. Hence, we have the following proposition.

Proposition 17. For a twin-free bipartite graph $G = (X, Y, E)$,

$$\chi_{NL}(G) \leq \min\{|X|, |Y|\} + 1.$$

6.2.1 Chain Graphs

In this subsection, we discuss the neighbor-locating chromatic number for the class of chain graphs. A chain graph $G = (X, Y, E)$ has an ordering $(x_1, x_2, \dots, x_{n_1}, y_1, y_2, \dots, y_{n_2})$ of vertices of G such that $N(x_1) \subseteq N(x_2) \subseteq \dots \subseteq N(x_{n_1})$ and $N(y_1) \supseteq N(y_2) \supseteq \dots \supseteq N(y_{n_2})$. Note that $X = \{x_1, x_2, \dots, x_{n_1}\}$ and $Y = \{y_1, y_2, \dots, y_{n_2}\}$. This ordering of vertices of G is called a chain ordering of G . Throughout this subsection, $G = (X, Y, E)$ denotes a chain graph.

The open twin relation provides a partition for both sides of the vertex set of G . Let $P_X = \{X_1, X_2, \dots, X_k\}$ and $P_Y = \{Y_1, Y_2, \dots, Y_k\}$ denotes the partition obtained for the X and Y side respectively. We have kept the order of the sets in P_X and P_Y such that they satisfy $N(X_1) \subset N(X_2) \subset \dots \subset N(X_k)$ and $N(Y_1) \supset N(Y_2) \supset \dots \supset N(Y_k)$. Note that $N(X_i) = \cup_{j=1}^i Y_j$ and $N(Y_i) = \cup_{j=i}^k X_j$ for each $i \in [k]$. Let $t = \max\{|X_i| : i \in [k]\}$ and $s = \max\{|Y_i| : i \in [k]\}$ for G .

Theorem 6.1. For a chain graph $G = (X, Y, E)$, the following holds true.

1. $\chi_{NL}(G) \leq 2k + s + t - 2$.

2. $\chi_{NL}(G) \leq \min\{|X| + s, |Y| + t\}$.
3. $\chi_{NL}(G) \geq \max\{s, t\} + 1$.
4. $\chi_{NL}(G) \geq |Y_1| + t$.
5. $\chi_{NL}(G) \geq |X_k| + s$.

Proof. We prove each stated bound one by one.

1. To show, $\chi_{NL}(G) \leq 2k + s + t - 2$, we give a NL-coloring of G using $2k + s + t - 2$ colors. Assign the color i to exactly one vertex of X_i for each $i \in [k]$. Next, assign the color $k + i$ to exactly one vertex of Y_i for each $i \in [k]$. Now, consider the set of colors $A = \{a_1, a_2, \dots, a_{t-1}\}$ and $B = \{b_1, b_2, \dots, b_{s-1}\}$. Assign a unique color from the set A to each uncolored vertex of X_i for each $i \in [k]$. Similarly, assign a unique color from the set B to each uncolored vertex of Y_i for each $i \in [k]$. We call this coloring c . We see that c is a proper coloring of G as we have assigned colors to the vertices of X from the set $[k] \cup A$ and to the vertices of Y from the set $\{k + 1, k + 2, \dots, 2k\} \cup B$. Note that c uses $2k + |A| + |B| = 2k + t - 1 + s - 1 = 2k + s + t - 2$ colors. Now, we need to show that c is a NL-coloring of G . For this, let u, v be two vertices of G having the same color. Now, there can be two cases: $u \in X_i, v \in X_j$ for some $i, j \in [k]$ or $u \in Y_i, v \in Y_j$ for some $i, j \in [k]$. In the former case, without loss of generality, we assume that $i < j$, then $k + i + 1 \in c(N(v))$ but $k + i + 1 \notin c(N(u))$, so $c(N(u)) \neq c(N(v))$. Similarly, in the latter case, without loss of generality, we assume that $i < j$, then $i \in c(N(u))$ but $i \notin c(N(v))$, so $c(N(u)) \neq c(N(v))$. Therefore, c is a NL-coloring of G which uses $2k + s + t - 2$ colors.
2. To show, $\chi_{NL}(G) \leq \min\{|X| + s, |Y| + t\}$, we show that $\chi_{NL}(G) \leq |X| + s$ and $\chi_{NL}(G) \leq |Y| + t$. First, we prove that $\chi_{NL}(G) \leq |X| + s$. For this, we give a NL-coloring of G using $|X| + s$ colors. Recall that $X = \{x_1, x_2, \dots, x_{n_1}\}$. Let B denotes the set $\{b_1, b_2, \dots, b_s\}$. Now, assign the vertex x_i of X the color i for $i \in [n_1]$ and assign a unique color from the set B to each uncolored vertex of Y_i for each $i \in [k]$. Clearly, this is a proper coloring since the set of colors used in X side is the set $[n_1]$ and the set of colors used in Y side is the set B . Now, let u, v be two vertices of the

same color, then $u \in Y_i$ and $v \in Y_j$ for some $i, j \in [k]$. If $i < j$, then $N(u) \subset N(v)$ and all vertices of X have different colors, so $c(N(u)) \neq c(N(v))$. Hence, we have a NL-coloring of G using $|X| + s$ colors. So, $\chi_{NL}(G) \leq |X| + s$. In a similar manner, we can assign colors to the vertices of G so that we have a NL-coloring of G which uses $|Y| + t$ colors. So, $\chi_{NL}(G) \leq |Y| + t$. Consequently, $\chi_{NL}(G) \leq \min\{|X| + s, |Y| + t\}$.

3. For the chain graph G , $\tau(G) = \max\{s, t\}$. So, $\chi_{NL}(G) \geq \max\{s, t\} + 1$ holds using Proposition 14.
4. Again, let X_i be the set such that $t = |X_i|$. Since all vertices of X_i are open twins, each vertex of X_i must get a different color in any NL-coloring of G , and for each $y \in Y_1$, $N(y) = X$, so vertices in Y_1 must receive distinct colors different from the colors given to the vertices of X . Thus, $\chi_{NL}(G) \geq |Y_1| + t$.
5. $\chi_{NL}(G) \geq |X_k| + s$ holds due to the similar arguments we have given in the proof of the previous bound.

□

Now, we prove the results providing the exact value of the neighbor-locating chromatic number of some restricted chain graphs. A neighbor-locating coloring of these chain graphs that uses the minimum number of colors is given in their respective proofs.

Theorem 6.2. *For a chain graph G , if $|X_1| > |X_2| > \dots > |X_k|$ and $|Y_i| \leq |Y_1|$ for each $i \in \{2, \dots, k\}$, then $\chi_{NL}(G) = |Y_1| + |X_1|$.*

Proof. Here, we observe that $t = |X_1|$ and $s = |Y_1|$, so $|Y_1| + |X_1| = s + t$. Using Theorem 6.1, we already know that $\chi_{NL}(G) \geq s + t$. Now, we give a NL-coloring of G which uses $s + t$ colors. Consider the sets $A = \{a_1, a_2, \dots, a_t\}$ and $B = \{b_1, b_2, \dots, b_s\}$ as two ordered sets of t and s colors respectively. Now, we define a coloring $c : V(G) \rightarrow A \cup B$ as follows. For $i \in [k]$, assign each vertex of X_i a unique color from the set A . It is possible since $|X_i| \leq t = |A|$ for each $i \in [k]$. Clearly, $c(X_1) = A$. We assign colors to the vertices of X_i , $i > 1$ in such a way that $c(X_i) = \{a_1, a_2, \dots, a_{t'}\}$, where $t' = |X_i|$. Now, we assign

colors to the vertices of Y . Color all vertices of Y_1 with a unique color from the set B . So, $c(Y_1) = B$.

Since $N(Y_2) \subset N(Y_1)$ and $|X_1| > |X_2|$, at least one color from the set A is not in the neighborhood of Y_2 . So, first we assign all such colors uniquely to the vertices of Y_2 and if there are some uncolored vertex in Y_2 , we give all of them a unique color from the set B . In a similar manner, color all vertices of $\cup_{j=3}^k Y_j$ in the order (Y_3, \dots, Y_K) . Thus, we have a proper coloring in G using $s + t$ colors. Next, we show that c is a NL-coloring of G . For this, let $u, v \in V(G)$ be two vertices of the same color in the coloring c . We have three cases to consider.

Case 1: $u \in X_i$ and $v \in X_j$ for some $i, j \in [k]$, $i < j$:

In this case, we see that there exists a vertex in Y_{i+1} whose color is not given to any of the vertices in the set $\cup_{j=1}^i Y_j$ which is equal to the set $N(X_i)$. So, $c(N(u)) \neq c(N(v))$.

Case 2: $u \in Y_i$ and $v \in Y_j$ for some $i, j \in [k]$, $i < j$:

In this case, we see that there exists a vertex in X_i whose color is not given to any of the vertices in the set $\cup_{j=i+1}^k X_j$ which is a superset of the set $N(Y_j)$. So, $c(N(u)) \neq c(N(v))$.

Case 3: $u \in X_i$ and $v \in Y_j$ for some $i, j \in [k]$, $i < j$:

In this case, we see that $B \subseteq c(N(X_i))$ as $c(Y_1) = B$ but $B \cap c(N(Y_j)) = \emptyset$. So, $c(N(u)) \neq c(N(v))$.

Hence, c is a NL-coloring of G which uses $s + t$ colors. Therefore, $\chi_{NL}(G) = s + t = |Y_1| + t$. □

The proof of the Theorem 6.3 is similar to the proof of the Theorem 6.2 and, hence is omitted.

Theorem 6.3. *For a chain graph G , if $|Y_1| < |Y_2| < \dots < |Y_k|$ and $|X_i| \leq |X_k|$ for each $i \in [k - 1]$, then $\chi_{NL}(G) = |X_k| + s$.*

Theorem 6.4. *Let $G = (X, Y, E)$ be a chain graph such that $N(x_1) \subset N(x_2) = \dots = N(x_{n_1})$, $N(y_1) = \dots = N(y_{i-1}) \supset N(y_i) = \dots = N(y_{n_2})$ for some $i \leq p$ and $n_1 = n_2 = p$ then*

1. *If $p > 2i - 1$, then $\chi_{NL}(G) = 2p - i$.*

2. If $p \leq 2i - 1$, then $\chi_{NL}(G) = p + i - 1$.

Proof. Suppose that c is a NL-coloring of G using $\chi_{NL}(G)$ colors implying $|c(V(G))| = \chi_{NL}(G)$. Note that $k = 2$, $X_1 = \{x_1\}$, $X_2 = \{x_2, \dots, x_p\}$, $Y_1 = \{y_1, \dots, y_{i-1}\}$ and $Y_2 = \{y_i, \dots, y_p\}$ for the graph G . Since $t = |X_2| = p - 1$ and $|Y_1| = i - 1$, $\chi_{NL}(G) \geq p + i - 2$ using Theorem 6.1. We observe that all vertices of $X_2 \cup Y_1$ have different colors in any NL-coloring of G , so $|c(X_2 \cup Y_1)| = p + i - 2$. Now, we have two cases to consider.

Case 1: $c(x_1) \in c(X_2)$:

In this case, we observe that $c(y) \notin c(X \cup Y_1)$ for each $y \in Y_2$. So, $\chi_{NL}(G) = p + i - 2 + |Y_2| = p + i - 2 + p - i + 1 = 2p - 1$.

Case 2: $c(x_1) \notin c(X_2)$:

In this case, $c(x_1) \notin c(X_2 \cup Y_1)$. So, $|c(X \cup Y_1)| = p + i - 1$. Now, either $|Y_2| > |Y_1| + 1$ or $|Y_2| \leq |Y_1| + 1$. If $|Y_2| > |Y_1| + 1$, that is, $p > 2i - 1$, at least $|Y_2| - (|Y_1| + 1) = p - i + 1 - i + 1 - 1 = p - 2i + 1$ more colors are used in the coloring c . So, $\chi_{NL}(G) = p + i - 1 + p - 2i + 1 = 2p - i$. Otherwise, if $|Y_2| \leq |Y_1| + 1$, that is, $p \leq 2i - 2$, colors of all vertices of Y_2 can coincide with the colors of $Y_1 \cup X_1$. So, $\chi_{NL}(G) = |c(X \cup Y_1)| = p + i - 1$.

As $2p - 1 \geq \max\{2p - i, p + i - 1\}$, we get, $\chi_{NL}(G) = 2p - i$ when $p > 2i - 1$ and $\chi_{NL}(G) = p + i - 1$ when $p \leq 2i - 1$. Therefore, the theorem holds. \square

6.2.2 Proper Interval Graphs

In this subsection, we discuss bounds on the neighbor-locating chromatic number of a proper interval graph G in terms of independence number and clique number of G . For shorthand, we write “proper interval graph” as *PIG*. Let G be a *PIG* and $\pi = (v_1, v_2, \dots, v_n)$ be a *BCO* of vertices of G . Note that for $i < j$, v_i appears before v_j in the ordering π . We write $v_i <_\pi v_j$ to denote that v_i appears before v_j in the ordering π . We define *last neighbor* of a vertex $v_i \in V(G)$, denoted by $l(v_i)$, as the highest indexed neighbor of v_i in the ordering π . The following observation can be proved using the properties of a *BCO* of G .

Observation 7. If $v_i v_j \in E(G)$ and $i < j$, then for each k , $i < k < j$; we have $v_i v_k, v_k v_j \in E(G)$.

Recall that $\alpha(G)$ denotes the independence number of G , $\chi(G)$ denotes the chromatic number of G and $\omega(G)$ denotes the clique number of G . Theorem 6.5 gives a relation between $\chi(G)$ and $\omega(G)$ for an interval graph [11].

Theorem 6.5. *For an interval graph G , $\chi(G) = \omega(G)$.*

Since every PIG is an interval graph, the above theorem also holds for proper interval graphs. Now, we are ready to prove the main theorem of this section.

Theorem 6.6. *For a proper interval graph G , $\chi_{NL}(G) \leq \alpha(G) + \omega(G) - 1$. Moreover, the bound is tight.*

Proof. First, we assume that $G \not\cong K_n$. We need to show that $\chi_{NL}(G) \leq \alpha(G) + \omega(G) - 1$. So, we give a NL-coloring of G which uses exactly $\alpha(G) + \omega(G) - 1$ colors.

First, we find a maximal independent set I of G containing the vertex v_1 . Let $I = \{v_{i_1}, v_{i_2}, v_{i_3}, \dots, v_{i_t}\}$, where $v_{i_1} = v_1$ and $v_{i_j} <_{\pi} v_{i_{j'}}$ for each $j < j'$. We find such an independent set I in t steps as follows. In the first step, $I = \{v_1\}$. Now, for each i from 2 to t , we pick a vertex $v_k \in V(G)$ such that v_k is the smallest indexed vertex that is not adjacent to the vertex included in the previous step and we include v_k in I . We stop when no more vertex can be included in I .

Now, we give a coloring c to the vertices of G in the following way. We assign colors in two stages. In stage 1, we assign a unique color to each vertex of I . By doing this, we have used $|I|$ colors in stage 1 and $|I| \leq \alpha(G)$. Next, we observe that exactly one vertex of each maximal clique is colored in stage 1. In stage 2, we color all the remaining uncolored vertices of G . Using Theorem 6.5, we know that $\chi(G) = \omega(G)$ and exactly one vertex of each maximal clique is colored in stage 1, so there exists a proper coloring of G , say c' in which only $\omega(G) - 1$ colors are used while coloring vertices of $V(G) \setminus I$. For a vertex $u \in V(G) \setminus I$, $c(u) = c'(u)$. We can ensure that $c(I) \cap c(V(G) \setminus I) = \emptyset$. Thus, we have colored each vertex of G and $|c(V(G))| \leq \alpha(G) + \omega(G) - 1$. As c' is a proper coloring that is used in stage 2 and all colors assigned in stage 1 are different from the colors used in stage 2, c is also a proper coloring of G . Now, we show that c is a NL-coloring of G . Let

$v_i, v_j \in V(G)$, $i < j$ such that $c(v_i) = c(v_j)$. This implies that $v_i v_j \notin E(G)$ and both of these vertices are colored in stage 2. So, $v_i, v_j \notin I$. Since I is a maximal independent set of G , there exists a vertex in I which is adjacent to v_i in G . Similarly, there exists a vertex in I which is adjacent to v_j in G . There are two cases to consider.

Case 1: $\exists w \in I$ such that w is adjacent to only one of v_i or v_j .

Without loss of generality, suppose that $\exists w \in I$ such that $w v_i \in E(G)$ and $w v_j \notin E(G)$. Since w is colored in stage 1, the set $\{a \in V(G) : c(a) = c(w)\}$ is equal to $\{w\}$. So, we have, $c(w) \in c(N(v_i))$ but $c(w) \notin c(N(v_j))$. Hence, $c(N(v_i)) \neq c(N(v_j))$.

Case 2: $\nexists w \in I$ such that w is adjacent to only one of v_i or v_j .

Let $w \in I$ be a vertex such that $w v_i, w v_j \in E(G)$. Note that $v_i < w < v_j$, since otherwise v_i and v_j must be adjacent and can not receive the same color. Moreover, we get, $n \geq 4$. Recall that $v_i v_j \notin E(G)$. Since v_i is colored in stage 2, there exists a vertex say v_k ($k < i$) such that v_k is colored in stage 1 and $v_k v_i \in E(G)$. As $v_k \in I$, we must have, $v_k w \notin E(G)$ implying that $v_k v_j \notin E(G)$. This gives a contradiction to our assumption that there is no vertex in G which is adjacent to only one of v_i or v_j . Hence, case 2 will not arise.

Thus, c is a NL-coloring of G . So, $\chi_{NL}(G) \leq \alpha(G) + \omega(G) - 1$. Now, if $G \cong K_n$ then $\chi_{NL}(G) = n = 1 + n - 1 = \alpha(G) + \omega(G) - 1$. Therefore, the bound holds for any FIG. The bound is tight for the class of complete graphs. \square

6.2.3 Co-bipartite Graphs

In this subsection, we discuss the neighbor-locating chromatic number of co-bipartite graphs. A graph is a co-bipartite graph if its complement is a bipartite graph. In other words, a co-bipartite graph is a graph whose vertex set can be partitioned into two sets V_1 and V_2 such that $G[V_1]$ and $G[V_2]$ are complete subgraphs of G . We call the subsets V_1 and V_2 , a co-bipartition of $V(G)$. Throughout this subsection, G denotes a co-bipartite graph with co-bipartition $\{V_1, V_2\}$. Let $|V_1| = n_1$ and $|V_2| = n_2$. Without loss of generality, we assume that $n_1 \geq n_2$. Since $G[V_1] \cong K_{n_1}$, we have, $\chi_{NL}(G) \geq n_1$ as all vertices of V_1 must be assigned different colors in any NL-coloring of G . We also have, $\chi_{NL}(G) \leq n_1 + n_2$.

Recall that a matching M of a graph G is a set of independent edges of G . Suppose M is a matching of \overline{G} and c is a proper coloring of $G[V_1]$. Note that each vertex of V_1 must have

received a unique color in the coloring c as $G[V_1] \cong K_{n_1}$. We denote the color assigned to the vertex $v \in V_1$ by c_v . Now, using the matching M of \overline{G} and the proper coloring c of $G[V_1]$, we define a proper coloring for the graph G . For this, we need to assign colors to each vertex of V_1 and V_2 . First, we assign the color c_v to each vertex $v \in V_1$. Then we assign colors to the vertices of V_2 . For this purpose, we consider the matching M of \overline{G} . Each edge $e \in M$ is of the form xy , where $x \in V_1$ and $y \in V_2$. Let xy be such an edge, then we assign the vertex $y \in V_2$ the color c_x . For all the remaining vertices of V_2 , we assign a different color uniquely. In this way, we have assigned colors to all the vertices of G and it is a proper coloring of G . Thus, given a proper coloring on the vertices of $G[V_1]$ and a matching of \overline{G} , we can define a proper coloring for all vertices of G . We call the coloring of G defined in such a way, the coloring M_c , where M denotes the given matching and c denotes the coloring assigned to the vertices of V_1 .

Now, we state a result that describes the neighbor-locating chromatic number of restricted kind of co-bipartite graphs.

Theorem 6.7. *Let G be a co-bipartite graph such that there exists a vertex $u \in V_2$ with $N(u) \cap V_1 = \emptyset$. If there exists a matching of \overline{G} saturating all vertices of V_2 then $\chi_{NL}(G) \in \{n_1, n_1 + 1\}$.*

Proof. Let $\mathcal{M} = \{M : M \text{ is a matching of } \overline{G} \text{ saturating all vertices of } V_2\}$. Note that $\mathcal{M} \neq \emptyset$. Suppose c is a proper coloring for the vertices of $G[V_1]$. Since $G[V_1] \cong K_{n_1}$, we assume that each vertex of V_1 is assigned a unique color from the set $[n_1]$ in the coloring c . For each $M \in \mathcal{M}$, consider the proper coloring M_c of G which assigns colors to all vertices of G .

Now, either there exists a matching $M^* \in \mathcal{M}$ such that the proper coloring M_c^* is a neighbor-locating coloring of G or there is no such matching. In the former case, $\chi_{NL}(G) = n_1$. In the latter case, $\chi_{NL}(G) \geq n_1 + 1$.

Next, we prove that $\chi_{NL}(G) = n_1 + 1$ if M_c is not a neighbor-locating coloring of G for each $M \in \mathcal{M}$. For this, we give a NL-coloring of G that uses $n_1 + 1$ colors. According to the statement of the theorem, there exists a vertex $u \in V_2$ such that u is not adjacent to any vertex of V_1 in G . So, $n_2 \geq 2$ as G is connected. Let $M \in \mathcal{M}$ which implies that it saturates

each vertex of V_2 . Note that there exists an edge $e \in M$ that saturates the vertex u . We see that the set $M' = M \setminus \{e\}$ is non-empty and is also a matching of \overline{G} . Now, in the coloring M'_c , the vertex $u \in V_2$ is assigned a different color from the colors used for the vertices of V_1 . So, the number of colors used in the coloring M'_c is $n_1 + 1$. In the coloring M'_c , the color assigned to the vertex u is not in the neighborhood of any vertex of V_1 and it is present in the neighborhoods of all the vertices of $V_2 \setminus \{u\}$. Thus, we get that M'_c is a NL-coloring of G and hence, $\chi_{NL}(G) = n_1 + 1$. \square

Now, we define some terminologies which we require for the rest of this subsection. For sets $A \subseteq V_1$, $B \subseteq V_2$, the set of edges $\{uv \in E(\overline{G}) : u \in A, v \in B\}$ is denoted by $E_{AB}(\overline{G})$. The set $V_1 \setminus A$ is denoted by A' and the set $V_2 \setminus B$ is denoted by B' .

Let $S_2 \subseteq V_2$. We say that S_2 satisfies property P if there exists a non-empty set $S_1 \subseteq V_1$ such that $|S_1| = n_1 - n_2 + |S_2|$ and the following holds: (i) for each $x \in S_1$, $|\{y \in S_2 : y \notin N(x)\}| \leq 1$ and, (ii) for each $y \in S_2$, $|\{x \in S_1 : x \notin N(y)\}| \leq 1$. Suppose $S_2 \subseteq V_2$ satisfies the property P then the set S_1 is called a P -pair of S_2 . The set of all P -pairs of S_2 is denoted by PS_2 .

We define sets Y and Z for a co-bipartite graph G as follows. $Y = \{S_2 \subseteq V_2 : PS_2 \neq \emptyset\}$, $Z = \{S_2 \in Y : \exists \text{ a matching } M \text{ of } \overline{G}[S'_1 \cup S'_2] \text{ that saturates each vertex of } S'_2 \text{ for some } P\text{-pair } S_1 \text{ of } S_2 \text{ and } \overline{G}[M \cup E_{S_1 S'_2}(\overline{G}) \cup E_{S'_1 S_2}(\overline{G})] \text{ contains no isolated edge}]\}$.

Finally, we state the theorem which exactly describes the neighbor-locating chromatic number of a co-bipartite graph G .

Theorem 6.8. *For a co-bipartite graph G , $\chi_{NL}(G) = n_1 + r$, where $r = \min\{|S_2| : S_2 \in Z\}$.*

Proof. First, we show that $\chi_{NL}(G) \geq n_1 + r$. On the contrary, assume that $\chi_{NL}(G) < n_1 + r$. So, let $\chi_{NL}(G) = n_1 + k$, where $k < r$. Let c be a NL-coloring of G which uses $\chi_{NL}(G)$, that is, $n_1 + k$ colors. Suppose $\{a_1, a_2, \dots, a_{n_1}, b_1, b_2, \dots, b_k\}$ is the set of $n_1 + k$ colors used in the coloring c . For notational convenience, we write $C_1 = \{a_1, a_2, \dots, a_{n_1}\}$ and $C_2 = \{b_1, b_2, \dots, b_k\}$. We also assume that $c(V_1) = C_1$ and $C_2 \subseteq c(V_2)$.

Let $S_2 = \{y \in V_2 : c(y) \in C_2\}$ and $S_1 = \{x \in V_1 : c(x) \notin c(V_2)\}$, then we see that S_1 is a P -pair of S_2 . Hence, $S_2 \in Y$. We claim that $S_2 \in Z$. To see this, note that, for each vertex $x \in S'_1$, there is exactly one vertex $y \in S'_2$ such that $c(x) = c(y)$. Now, the set $M = \{xy : c(x) = c(y)\}$ is a matching of $\overline{G}[S'_1 \cup S'_2]$ that saturates each vertex of S'_2 . Since c is a NL-coloring of G , $c(N(x)) \neq c(N(y))$ for every pair of vertices x, y of G satisfying $c(x) = c(y)$. Let $xy \in M$ implying that $c(x) = c(y)$, then $c(N(x)) \neq c(N(y))$. This means that there is a color in the set C_2 which is not in the set $c(N(x))$ or there is a color in the set $c(S_1)$ which is not in the set $c(N(y))$, that is, $c(N(x)) \cap C_2 \neq C_2$ or $c(N(y)) \cap c(S_1) \neq c(S_1)$. Thus, for each edge in M , there exists an edge adjacent to it in $E_{S_1 S'_2}(\overline{G}) \cup E_{S'_1 S_2}(\overline{G})$. Thus, $S_2 \in Z$. So, $|S_2| \geq r$ as $r = \min\{|S_2| : S_2 \in Z\}$ which further implies that $k \geq r$, a contradiction. So, $\chi_{NL}(G) \geq n_1 + r$.

Next, we give a NL-coloring of G that uses $n_1 + r$ colors. Let $S_2 \in Z$ such that $|S_2| = r$. Now, assign a unique color to each vertex of V_1 from the set $[n_1]$. Denote this coloring of $G[V_1]$ by c_0 . Since $S_2 \in Z$, there exists a matching M of $\overline{G}[S'_1 \cup S'_2]$ that saturates each vertex of S'_2 , where S_1 is a P -pair of S_2 . As all vertices of V_1 are properly colored by the coloring c_0 and M is a matching of \overline{G} , there exists a proper coloring M_{c_0} for the whole vertex set of G . We assign each vertex of V_2 the colors assigned by M_{c_0} , hence all vertices of G are properly colored and the number of colors used is $n_1 + r$. To see that it is also a NL-coloring, let $x \in S'_1$ and $y \in S'_2$ be the vertices such that $M_{c_0}(x) = M_{c_0}(y)$. Since $S_2 \in Z$, there exists a vertex $x' \in S_1$ such that $x'y \notin E(G)$ or there is a vertex $y' \in S_2$ such that $xy' \notin E(G)$. So, $M_{c_0}(N(x)) \neq M_{c_0}(N(y))$. Hence, M_{c_0} is a NL-coloring of G . Therefore, $\chi_{NL}(G) = n_1 + r$. \square

Now, using the Theorem 6.8 we characterize all co-bipartite graphs whose neighbor-locating chromatic number is n_1 or $n_1 + n_2$.

Theorem 6.9. $\chi_{NL}(G) = n_1$ if and only if the following holds: (i) $n_1 > n_2$, (ii) there exists a set $S_1 \subseteq V_1$ with $|S_1| = n_1 - n_2$ such that, for each $S_2 \subseteq V_2$, $|S_2| \leq |N_{\overline{G}}(S_2) \cap S'_1|$ and, (iii) the set of edges $E_{S_1 V_2}(\overline{G})$ covers all vertices of V_2 .

Theorem 6.10. $\chi_{NL}(G) = n_1 + n_2$ if and only if for every edge $xy \in E(\overline{G})$ ($x \in V_1, y \in V_2$), we have $xy' \in E(G)$ for every $y' \in V_2 \setminus \{y\}$ and $x'y \in E(G)$ for every $x' \in V_1 \setminus \{x\}$.

The proof of both the Theorems 6.9 and 6.10 follow from the Theorem 6.8.

6.3 Approximation Algorithm

In this section, we present an approximation algorithm for the NLC problem in general graphs. For basic definitions related to approximation algorithms, we refer [90].

To design the algorithm, we use a vertex cover of the given graph G . The VERTEX COVER problem asks to compute a minimum vertex cover of a given graph G . The VERTEX COVER problem is a known NP-complete problem [39]. There exists a 2-approximation algorithm for the VERTEX COVER problem in general graphs that works as follows: Pick an arbitrary edge, take both of its endpoints into vertex cover, and remove these two vertices from the graph. Repeat this until we have no edge in the graph. We name this algorithm ALGOVC for this section.

Now, we give an approximation algorithm (Algorithm 16) to compute a NL-coloring for a graph G . Note that we are considering connected graphs only.

Algorithm 16: An approximation algorithm: NL-coloring of a graph

Input: A connected graph $G = (V, E)$.

Output: a NL-coloring of G .

1. Find a vertex cover of the graph, say C using ALGOVC.
 2. Give each vertex of C a unique color from the set $[2\beta(G)]$.
 3. Find a partition of the vertices of $V \setminus C$ such that, if two vertices of $V \setminus C$ are open twins, keep them in the same part. Call that partition P .
 4. For each set $X \in P$, do the following:
Assign a unique color to each vertex of X from the set $\{2\beta(G) + 1, 2\beta(G) + 2, \dots, 2\beta(G) + \tau(G)\}$.
 5. Return the coloring.
-

Theorem 6.11 proves the correctness and ratio of the Algorithm 16.

Theorem 6.11. *Algorithm 16 is an $O(\beta(G))$ -approximation algorithm for the NLC problem in general graphs and it runs in linear-time.*

Proof. Let c be the coloring returned by the Algorithm 16. We can prove that c is a NL-coloring of G using the similar arguments as given in the proof of the Proposition 15. Now, we prove the ratio of the algorithm. Let $Algo(G)$ denote the number of colors used in the coloring c and $Opt(G) = \chi_{NL}(G)$. Then,

$$\begin{aligned}
 \frac{Algo(G)}{Opt(G)} &\leq \frac{2\beta(G) + \tau(G)}{\chi_{NL}(G)} \\
 &\leq \frac{2\beta(G) + \tau(G)}{1 + \tau(G)} \\
 &\leq \frac{2\beta(G) + \tau(G)}{\tau(G)} \\
 &\leq 1 + \frac{2\beta(G)}{\tau(G)} \\
 &\leq 1 + 2\beta(G)
 \end{aligned}$$

Note that each step of the Algorithm 16 can be implemented in linear-time. Hence, the theorem is proved. \square

6.4 Neighbor-Locating Coloring vs Vertex Coloring

Recall that the VERTEX COLORING problem asks to find a proper coloring of G using the minimum number of colors. In this section, we remark the complexity difference between the VERTEX COLORING problem and the NLC problem.

First, we state some observations as propositions in which we characterize all graphs having neighbor-locating chromatic numbers 1 or 2. The proofs of the Propositions are easy and hence, proofs are omitted.

Proposition 18. For a connected graph G , $\chi_{NL}(G) = 1$ if and only if $G \cong K_1$.

Proposition 19. For a connected graph G , $\chi_{NL}(G) = 2$ if and only if $G \cong K_2$ or $G \cong \overline{K_2}$.

Using Propositions 18 and 19, we can directly state that $\chi_{NL}(G) \geq 3$, for all connected graphs with $|V(G)| \geq 3$. Below, we state a result that has been proved in [3].

Theorem 6.12. *For a graph G with no isolated vertices, $|V(G)| \leq k(2^{k-1} - 1)$, where $k = \chi_{NL}(G)$.*

If G admits a proper coloring with at most k colors, we say that G is k -colorable. The k -COLORABILITY problem ($k \geq 2$) asks to find a k -coloring of a given graph G if G is k -colorable. Dailey et al. proved that the 3-COLORING problem is NP-complete [28]. In a similar manner, we can define the NEIGHBOR-LOCATING 3-COLORABILITY (NL3C) problem as follows: Given a graph G , find a NL-coloring of G using at most 3 colors, if such a NL-coloring exists. By NL - k -coloring, we mean a NL-coloring that uses k colors. Below, we prove that the NL3C problem can be solved in polynomial-time.

Theorem 6.13. *The NEIGHBOR-LOCATING 3-COLORABILITY problem is in class P.*

Proof. Let $G = (V, E)$ be the input graph for the NL3C problem and $n = |V(G)|$. To prove the theorem, we provide a polynomial-time algorithm that outputs a NL-3-coloring of G if such a coloring exists, otherwise, it returns that no such coloring exists. We also assume that G is a connected graph. Theorem 6.12 tells that $n \leq k(2^{k-1} - 1)$, where $k = \chi_{NL}(G)$. For $k = 3$, $k(2^{k-1} - 1) = 9$. Thus, if $n \geq 10$, it can never have a NL-coloring using 3 colors. So, let $n \leq 9$. For a graph with at most 9 vertices, NL-3-coloring can be computed by brute force method if it exists. Below, we further mention the details.

Note that a proper coloring of G that uses t colors, partitions the vertex set of G in t parts where all vertices in one part must have the same color. A neighbor-locating coloring is also proper coloring. So, a NL- t -coloring also partitions the vertex set in t parts with some additional requirements. Suppose c is a NL-coloring of G using 3 colors in which vertices $u, v \in V$ have been assigned the same color. Since c is a NL-coloring, we have, $c(N(u)) \neq c(N(v))$. We assume that 3 colors used in the coloring c are labeled as c_1, c_2 and c_3 . Without loss of generality, let $c(u) = c(v) = c_1$. This means that the set of colors assigned to the neighborhood of the vertices u, v must be a nonempty subset of $\{c_2, c_3\}$. As the number of nonempty subsets of $\{c_2, c_3\}$ are 3, we can say that 4 vertices of G must not get the same color in any NL-3-coloring of G . Thus, a NL-3-coloring of G corresponds to a positive solution of the equation $x_1 + x_2 + x_3 = n$, $1 \leq x_i \leq 3$. Note that the number of

positive solutions of the equation $x_1 + x_2 + x_3 = n, 1 \leq x_i \leq 3$ are $\sum_{n=3}^9 \binom{n-1}{2} = 84$. This suggests a simple algorithm for solving the NL3C problem in G .

Suppose a_1, a_2, a_3 is a solution of this equation. Then there are $\binom{n}{a_1} \cdot \binom{n-a_1}{a_2} \cdot \binom{n-(a_1+a_2)}{a_3}$ number of ways in which we can partition the set V into 3 parts such that the three parts contain a_1, a_2 and a_3 number of vertices, respectively. So, by looking at all possible such partitions, we can check in polynomial-time whether that partition leads to a neighbor-locating coloring of G . We describe the steps below.

1. If $n \geq 10$, output “No such coloring exists”.
2. If $n \leq 9$, find all positive solutions of the equation $x_1 + x_2 + x_3 = n, 1 \leq x_i \leq 3$.
3. For each solution $(x_1, x_2, x_3) = (a_1, a_2, a_3)$, find all ways of partitioning V into 3 parts containing a_1, a_2 and a_3 number of vertices respectively.
4. For each way of partitioning, check whether that leads to a NL-coloring or not.
5. If yes, return that partition as the coloring. Otherwise, output “No such coloring exists”.

It is easy to see that all these tasks can be performed in constant time. So, The NEIGHBOR-LOCATING 3-COLORABILITY problem is in class P . \square

For $k \geq 3$, the NEIGHBOR-LOCATING k -COLORABILITY (NL k C) problem is to find a NL-coloring of a given graph G using at most k colors, if such a coloring exists. We observe that finding such a coloring is equivalent to partitioning the vertex set of the graph in k parts satisfying certain conditions. The number of ways for doing such a partition is $O(n^{f(k)})$, where $f(k)$ represents a polynomial function of k . So, we can directly state the following result.

Theorem 6.14. *For any constant $k(k \geq 3)$, the NL k C problem is in class P .*

6.5 Summary

In this chapter, we proved some new bounds for the neighbor-locating chromatic number in general graphs as well as in some restricted graph classes. We also designed an approximation algorithm that runs in linear-time but the ratio is not constant. Although finding whether a graph G can be colored properly using 3 colors is a well-known NP-hard problem [28], we found that the similar problem in the neighbor-locating coloring turns out to be in class P.

Chapter 7

Conclusion and Future Directions

7.1 Contributions

In this thesis, we studied the following five graph optimization problems. We obtained numerous algorithmic and combinatorial results for these graph parameters.

1. MAXIMUM INTERNAL SPANNING TREE Problem
2. MINIMUM EDGE TOTAL DOMINATING SET Problem
3. GRUNDY (DOUBLE) DOMINATION Problem
4. MAXIMUM WEIGHTED EDGE BICLIQUE Problem
5. NEIGHBOR-LOCATING COLORING Problem

In Chapter 2, we studied the MIST problem which is a generalization of the HAMILTONIAN PATH problem. Since the HAMILTONIAN PATH problem is NP-hard for chordal and bipartite graphs, the MIST problem also remains NP-hard for chordal and bipartite graphs [54, 71]. We studied the MIST problem for the following classes of graphs: chain graphs, bipartite permutation graphs, block graphs, cactus graphs and cographs. We found linear-time algorithms for the MIST problem for each of these graph classes.

Li et al. [59] proved an upper bound for $Opt(G)$ in terms of $|E(P^*)|$, where $Opt(G)$ denotes the number of internal vertices in a MIST of G and $|E(P^*)|$ denotes the number of edge in an optimal path cover of G . We further studied the relationship between the number of edges in optimal path covers and $Opt(G)$ and provided tight lower bounds for chain graphs and cographs. We also proved that this phenomenon does not hold for general graphs with the construction of bipartite permutation graphs and block graphs for which $Opt(G)$ is arbitrarily far from $|E(P^*)|$.

In Chapter 3, we studied the computational complexity of the Min-ETDS problem.

We discussed the complexity difference between the Min-EDS and the Min-ETDS problem and proved that they differ in terms of complexity. We resolved the complexity status of the problem in chain graphs. Further, we studied the problem in two subclasses of chordal graphs which are split graphs and proper interval graphs without any cut vertices.

We also studied the approximation aspect of this problem. We proved that the problem is APX-complete for graphs with maximum degree 3 and designed an efficient approximation algorithm for k -regular graphs when $k \geq 4$. We remark that our approximation algorithm is based on a relationship between the cardinality of min-ETD-set and min-CVC of a graph.

The next chapter, that is, Chapter 4 is devoted to the vertex sequences in graphs. Precisely, we discussed the Grundy dominating and Grundy double dominating sequences in graphs. We showed that the GDD problem is NP-complete for bipartite and co-bipartite graphs and linear-time solvable for chain graphs which is a subclass of bipartite graphs. For the GD2D problem, we proved that it also remains NP-complete for split, bipartite and co-bipartite graphs. On the positive side, we designed linear-time algorithms for threshold graphs and chain graphs.

In Chapter 5, we studied the MWEB problem. We proved that the decision version of the MWEB problem remains NP-complete even for complete bipartite graphs, which is a subclass of bipartite graphs. On the positive side, we proved that by adding a restriction that if the weight of each edge is a positive real number, the MWEB problem is $O(n^2)$ -time solvable for bipartite permutation graphs and $O(m + n)$ -time solvable for chain graphs.

In Chapter 6, we studied the NLC problem. We gave certain bounds for the associated parameter in the following graph classes: chain graphs, proper interval graphs, and co-bipartite graphs. We also presented an approximation algorithm for general graphs. Further, we have presented a result showing that the complexity of the NLC problem differs from the VERTEX COLORING problem.

7.2 Future Directions

Below, we give some promising research directions for the five problems that we studied in the thesis.

- **MAXIMUM INTERNAL SPANNING TREE Problem**

A convex bipartite graph G with bipartition (X, Y) and an ordering $X = (x_1, x_2, \dots, x_n)$, is a bipartite graph such that for each $y \in Y$, the neighborhood of y in X appears consecutively. The complexity status of the MIST problem is still open for the class of convex bipartite graphs, which is a superclass of bipartite permutation graphs and a subclass of chordal bipartite graphs. We designed a polynomial algorithm for the MIST problem in bipartite permutation graphs. In bipartite graphs, the problem is already NP-complete. Designing an algorithm for the MIST problem in convex bipartite graphs will be a good research direction.

The weighted version of the MIST problem is also well studied in literature [81]. Given a vertex-weighted connected graph G , the MAXIMUM WEIGHT INTERNAL SPANNING TREE (MwIST) problem asks for a spanning tree T of G such that the total weight of internal vertices in T is maximized. Since the MwIST problem is a generalization of the MIST problem, one may also investigate the complexity status of the MwIST problem for some special classes of graphs.

As far as we know, every hardness proof in the literature for the MIST problem on families of graphs relies on a reduction to the HAMILTONIAN PATH problem. We leave as an open question if there exists a family of graphs such that the HAMILTONIAN PATH problem is polynomial time, but the MIST problem remains NP-hard.

- **MINIMUM EDGE TOTAL DOMINATING SET Problem**

We resolved the complexity status of the Min-ETDS problem in chain graphs. One may work on resolving the complexity status of the problem in other subclasses of bipartite graphs, sandwiched between bipartite graphs and chain graphs. Further, we studied the problem in proper interval graphs without a cut vertex. It would be interesting to

resolve the complexity of the problem in interval graphs.

If we look at the paper [26], it is shown that there is a 2-approximation algorithm to compute an edge dominating set of a given graph. So, a 4-approximation algorithm for the min-ETDS problem in general graphs is trivial. One may try to design an approximation algorithm with approximation ratio better than 4.

- **GRUNDY (DOUBLE) DOMINATION Problem**

For the GD and the GD2 problems, we resolved the complexity status in various special graph classes but there are still many graph classes in which the complexity status is unknown. Finding answers to these questions can be a good research direction. Note that the gap between NP-completeness and efficient algorithms of the GD and GD2 problems has come a bit closer in bipartite graphs, where our results from Chapter 4 show that, in bipartite graphs, both the problems are NP-complete. There is a polynomial-time algorithm for these problems in chain graphs. Noting the following inclusion relation between graph classes:

$$\text{Chain} \subsetneq \text{Bipartite Permutation} \subsetneq \text{Chordal Bipartite} \subsetneq \text{Bipartite},$$

the following question is natural, and its solution would narrow the gap a little more (in the bipartite case).

Problem 1. What is the computational complexity of determining the Grundy domination number and the Grundy double domination number in bipartite permutation graphs or in chordal bipartite graphs?

A similar, but somewhat larger gap comes from our results in Chapter 4 concerning the Grundy double domination. There are several classes of graphs that lie between the threshold graph (for which we found an efficient algorithm) and the split graph (for which we proved NP-completeness). Perhaps the most interesting class among these graph classes is strongly chordal split graphs. One may also investigate the complexity of Grundy double domination in cographs as for several Grundy domination-type invariants, efficient algorithm exists for cographs.

Problem 2. What is the computational complexity of determining the Grundy double domination number in strongly chordal split graphs (respectively cographs)?

Note that, due to the structure of co-chain graphs, we may infer that finding the Grundy domination number and the Grundy double domination number is easy for a co-chain graph. Recall that complement of a chain graph is a co-chain graph. So, it would be interesting to see if there are some other known classes of graphs \mathcal{G} whose complement class $\overline{\mathcal{G}} = \{\overline{G} : G \in \mathcal{G}\}$ has the similar status of the computational complexity of the GDD and the GD2D problem as \mathcal{G} . Two nice instances (with \mathcal{G} bipartite graphs and chain graphs) are presented in Section 4.2 and 4.3 of Chapter 4.

There is no result for the GD problem and the GD2 problem from the approximation point of view. One can also try to design an approximation algorithm for the problem. Several other types of vertex sequences were presented in [44] of which computational complexities have not yet been considered. The study of all those sequences is open from an algorithmic point of view.

- MAXIMUM WEIGHTED EDGE BICLIQUE Problem

It will be interesting to try to design a linear-time algorithm for the MWEB problem in bipartite permutation graphs, as for the unweighted case, this problem is linear-time solvable. One may also try to design a linear-time algorithm for the MEB problem in convex bipartite graphs. We have given efficient algorithms to solve the S -MWEB problem in some subclasses of bipartite graphs by taking the edge weights from the set of positive real numbers, that is, $S = \mathbb{R}^+$. For some different choices of S , finding the complexity status of the S -MWEB problem can be a good research direction.

- NEIGHBOR-LOCATING COLORING Problem

To continue the algorithmic study of the NLC problem, one can try to investigate the complexity status in various other graph classes. We strongly believe that the decision version of the NLC problem is NP-complete for general graphs. Below, we state this statement as a conjecture.

Conjecture 1. The decision version of the NLC problem is NP-complete for general graphs.

One may try to find the complexity status of the problem in some other graph classes such as bipartite graphs and chordal graphs. Although finding whether a graph G can be colored properly using 3 colors is a well-known NP-hard problem [28], we found that a similar problem in the neighbor-locating type of coloring comes out to be in class P. A useful research direction may be to find more results that compare the neighbor-locating coloring to other variations of vertex colorings.

For a co-bipartite graph G , one can try to answer the question that whether there is any polynomial time algorithm that computes an NL-coloring of G using $\chi_{NL}(G)$ colors. We designed an approximation algorithm that runs in linear-time but the ratio is not constant. This keeps the following question open: Does the NLC problem belong to the class APX?

Bibliography

1. Ahangar, H. A., Chellali, M., Sheikholeslami, S., Soroudi, M., and Volkmann, L. (2021). Total vertex-edge domination in trees. *Acta Mathematica Universitatis Comenianae*, 90(2):127–143.
2. Alcon, L., Gutierrez, M., Hernando, C., Mora, M., and Pelayo, I. M. (2019). The neighbor-locating-chromatic number of pseudotrees. *arXiv preprint arXiv:1903.11937*.
3. Alcon, L., Gutierrez, M., Hernando, C., Mora, M., and Pelayo, I. M. (2020). Neighbor-locating colorings in graphs. *Theoretical Computer Science*, 806:144–155.
4. Alcon, L., Gutierrez, M., Hernando, C., Mora, M., and Pelayo, I. M. (2021). The neighbor-locating-chromatic number of trees and unicyclic graphs. *Discussiones Mathematicae Graph Theory*.
5. Alexe, G., Alexe, S., Crama, Y., Foldes, S., Hammer, P. L., and Simeone, B. (2004). Consensus algorithms for the generation of all maximal bicliques. *Discrete Applied Mathematics*, 145(1):11–21.
6. Alimonti, P. and Kann, V. (1997). Hardness of approximating problems on cubic graphs. In *Italian Conference on Algorithms and Complexity*, pages 288–298. Springer.
7. Ambühl, C., Mastrolilli, M., and Svensson, O. (2007). Inapproximability results for sparsest cut, optimal linear arrangement, and precedence constrained scheduling. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 329–337. IEEE.
8. Ambühl, C., Mastrolilli, M., and Svensson, O. (2011). Inapproximability results for maximum edge biclique, minimum linear arrangement, and sparsest cut. *SIAM Journal on Computing*, 40(2):567–596.

9. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M. (1999). *Complexity and approximation*. Springer-Verlag, Berlin. Combinatorial optimization problems and their approximability properties.
10. Balakrishnan, R. and Ranganathan, K. (2000). *A textbook of graph theory*. Universitext. Springer-Verlag, New York.
11. Balakrishnan, R. and Ranganathan, K. (2012). *A textbook of graph theory*. Springer Science & Business Media.
12. Bell, K., Driscoll, K., Krop, E., and Wolff, K. (2021). Grundy domination of forests and the strong product conjecture. *arXiv preprint arXiv:2104.05665*.
13. Bertossi, A. (1984). Dominating sets for split and bipartite graphs. *Information Processing Letters*, 19:37–40.
14. Booth, K. and Johnson, J. (1982). Dominating sets in chordal graphs. *SIAM J. Comput.*, 11:191–199.
15. Booth, K. S. and Lueker, G. S. (1976). Testing for the consecutive ones property, interval graphs, and graph planarity using *PQ*-tree algorithms. *J. Comput. System Sci.*, 13(3):335–379.
16. Boutrig, R. and Chellali, M. (2018). Total vertex-edge domination. *International Journal of Computer Mathematics*, 95(9):1820–1828.
17. Brešar, B., Bujtás, C., Gologranc, T., Klavžar, S., Košmrlj, G., Patkós, B., Tuza, Z., and Vizer, M. (2016a). Dominating sequences in grid-like and toroidal graphs. *Electron. J. Combin.*, 23:4–34.
18. Brešar, B., Bujtás, C., Gologranc, T., Klavžar, S., Košmrlj, G., Patkós, B., Tuza, Z., and Vizer, M. (2017). Grundy dominating sequences and zero forcing sets. *Discrete Optimization*, 26:66–77.
19. Brešar, B., Gologranc, T., and Kos, T. (2016b). Dominating sequences under atomic changes with applications in sierpinski and interval graphs. *Appl. Anal. Discrete Math*, 10(2):518–531.

20. Brešar, B., Gologranc, T., Milanič, M., Rall, D. F., and Rizzi, R. (2014). Dominating sequences in graphs. *Discrete Mathematics*, 336:22–36.
21. Brešar, B., Henning, M. A., Klavžar, S., and Rall, D. F. (2021). *Domination games played on graphs*. Springer.
22. Brešar, B., Henning, M. A., and Rall, D. F. (2016c). Total dominating sequences in graphs. *Discrete Mathematics*, 339(6):1665–1676.
23. Brešar, B., Klavžar, S., and Rall, D. F. (2010). Domination game and an imagination strategy. *SIAM Journal on Discrete Mathematics*, 24(3):979–991.
24. Campêlo, M. and Severín, D. (2021). An integer programming approach for solving a generalized version of the grundy domination number. *Discrete Applied Mathematics*, 301:26–48.
25. Chen, Z.-Z., Harada, Y., Guo, F., and Wang, L. (2018). An approximation algorithm for maximum internal spanning tree. *Journal of Combinatorial Optimization*, 35(3):955–979.
26. Chlebík, M. and Chlebíková, J. (2006). Approximation hardness of edge dominating set problems. *Journal of Combinatorial Optimization*, 11(3):279–290.
27. Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT Press, Cambridge, MA, third edition.
28. Dailey, D. P. (1980). Uniqueness of colorability and colorability of planar 4-regular graphs are np-complete. *Discrete Mathematics*, 30(3):289–293.
29. Dawande, M., Keskinocak, P., Swaminathan, J. M., and Tayur, S. (2001). On bipartite and multipartite clique problems. *Journal of Algorithms*, 41(2):388–403.
30. Dawande, M., Keskinocak, P., and Tayur, S. (1997). On the biclique problem in bipartite graphs. *Tech. rep., Carnegie-Mellon University*.
31. Dias, V. M., De Figueiredo, C. M., and Szwarcfiter, J. L. (2005). Generating bicliques of a graph in lexicographic order. *Theoretical Computer Science*, 337(1-3):240–248.

32. Dias, V. M., de Figueiredo, C. M., and Szwarcfiter, J. L. (2007). On the generation of bicliques of a graph. *Discrete Applied Mathematics*, 155(14):1826–1832.
33. Downey, R. G. and Fellows, M. R. (2013). *Fundamentals of parameterized complexity*. Texts in Computer Science. Springer, London.
34. Erey, A. (2020). Uniform length dominating sequence graphs. *Graphs and Combinatorics*, 36(6):1819–1825.
35. Farber, M. (1984). Domination, independent domination, and duality in strongly chordal graphs. *Discrete Applied Mathematics*, 7:115–130.
36. Feige, U. (2002). Relations between average case complexity and approximation complexity. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 534–543. ACM.
37. Feige, U. and Kogan, S. (2004). Hardness of approximation of the balanced complete bipartite subgraph problem. *Dept. Comput. Sci. Appl. Math., Weizmann Inst. Sci., Rehovot, Israel, Tech. Rep. MCS04-04*.
38. Fulkerson, D. and Gross, O. (1965). Incidence matrices and interval graphs. *Pac. J. Math.*, 15:835–855.
39. Garey, M. R. and Johnson, D. S. (1979a). *Computers and intractability*, volume 174. freeman San Francisco.
40. Garey, M. R. and Johnson, D. S. (1979b). A guide to the theory of np-completeness. *Computers and intractability*, pages 641–650.
41. Gély, A., Nourine, L., and Sadi, B. (2009). Enumeration aspects of maximal cliques and bicliques. *Discrete applied mathematics*, 157(7):1447–1459.
42. Goerdt, A. and Lanka, A. (2004). An approximation hardness result for bipartite clique. In *Electronic Colloquium on Computational Complexity, Report*, volume 48.
43. Harary, F. (1970). Proof techniques in graph theory. Technical report, MICHIGAN UNIV ANN ARBOR DEPT OF MATHEMATICS.

44. Haynes, T. and Hedetniemi, S. (2021). Vertex sequences in graphs. *Discrete Math. Lett.*, 6:19–31.
45. Heggernes, P. and Kratsch, D. (2007). Linear-time certifying recognition algorithms and forbidden induced subgraphs. *Nord. J. Comput.*, 14(1-2):87–108.
46. Heggernes, P., Van't Hof, P., Lokshtanov, D., and Nederlof, J. (2012). Computing the cutwidth of bipartite permutation graphs in linear time. *SIAM Journal on Discrete Mathematics*, 26(3):1008–1021.
47. Hochbaum, D. S. (1998). Approximating clique and biclique problems. *Journal of Algorithms*, 29(1):174–200.
48. Jamison, R. E. and Laskar, R. (1982). Elimination orderings of chordal graphs. *Combinatorics and Applications*, pages 192–200.
49. Jung, H. A. (1978). On a class of posets and the corresponding comparability graphs. *Journal of Combinatorial Theory, Series B*, 24(2):125–133.
50. Karp, R. M. (1972). Reducibility among combinatorial problems, complexity of computer computations (re miller and jw thatcher, editors).
51. Kloks, T. and Kratsch, D. (1995). Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph. *Information Processing Letters*, 55(1):11–16.
52. Knauer, M. and Spoerhase, J. (2015). Better approximation algorithms for the maximum internal spanning tree problem. *Algorithmica*, 71(4):797–811.
53. Kulli, V. and Patwari, D. (1991). On the total edge domination number of a graph. In *Proc. of the Symp. On Graph Theory and Combinatorics, Kochi, Centre Math. Sci., Trivandrum, Series: Publication*, volume 21, pages 75–81.
54. Lai, T. H. and Wei, S. S. (1993). The edge hamiltonian path problem is np-complete for bipartite graphs. *Information processing letters*, 46(1):21–26.
55. Lai, T.-H. and Wei, S.-S. (1997). Bipartite permutation graphs with application to the minimum buffer size problem. *Discrete applied mathematics*, 74(1):33–55.

56. Lerchs, H. (1972). On the clique-kernel structure of graphs. *Dept. of Computer Science, University of Toronto*, 1.
57. Li, P., Shang, J., and Shi, Y. (2022). A simple linear time algorithm to solve the mist problem on interval graphs. *Theoretical Computer Science*, 930:77–85.
58. Li, W., Cao, Y., Chen, J., and Wang, J. (2017). Deeper local search for parameterized and approximation algorithms for maximum internal spanning tree. *Information and Computation*, 252:187–200.
59. Li, X., Feng, H., Jiang, H., and Zhu, B. (2018). Solving the maximum internal spanning tree problem on interval graphs in polynomial time. *Theoretical Computer Science*, 734:32–37.
60. Li, X. and Zhu, D. (2014). Approximating the maximum internal spanning tree problem via a maximum path-cycle cover. In *International symposium on algorithms and computation*, pages 467–478. Springer.
61. Li, X., Zhu, D., and Wang, L. (2021). A $4/3$ -approximation algorithm for the maximum internal spanning tree problem. *Journal of Computer and System Sciences*, 118:131–140.
62. Li, Y., Wang, W., and Yang, Z. (2019). The connected vertex cover problem in k -regular graphs. *Journal of Combinatorial Optimization*, 38(2):635–645.
63. Lin, J. C.-H. (2019). Zero forcing number, grundy domination number, and their variants. *Linear Algebra and its Applications*, 563:240–254.
64. Lin, R., Olariu, S., and Pruesse, G. (1995). An optimal path cover algorithm for cographs. *Computers & Mathematics with Applications*, 30(8):75–83.
65. Lu, H.-I. and Ravi, R. (1992). The power of local optimization: Approximation algorithms for maximum-leaf spanning tree. In *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, volume 30, pages 533–533. University of Illinois.
66. Mahadev, N. V. and Peled, U. N. (1995). *Threshold graphs and related topics*. Elsevier.

67. Malaguti, E. and Toth, P. (2010). A survey on vertex coloring problems. *International transactions in operational research*, 17(1):1–34.
68. Manurangsi, P. (2018). Inapproximability of maximum biclique problems, minimum k-cut and densest at-least-k-subgraph from the small set expansion hypothesis. *Algorithms*, 11(1):10.
69. Mojdeh, D. A. (2022). On the conjectures of neighbor locating coloring of graphs. *Theoretical Computer Science*.
70. Muddebihal, M. and Sedamkar, A. (2013). Characterization of trees with equal edge domination and end edge domination numbers. *International Journal of Mathematics and Computer Applications Research*, 3(1):33–42.
71. Müller, H. (1996). Hamiltonian circuits in chordal bipartite graphs. *Discrete Mathematics*, 156(1-3):291–298.
72. Nasini, G. and Torres, P. (2020). Grundy dominating sequences on x-join product. *Discrete Applied Mathematics*, 284:138–149.
73. Nussbaum, D., Pu, S., Sack, J.-R., Uno, T., and Zarrabi-Zadeh, H. (2012). Finding maximum edge bicliques in convex bipartite graphs. *Algorithmica*, 64(2):311–325.
74. Pak-Ken, W. (1999). Optimal path cover problem on block graphs. *Theoretical computer science*, 225(1-2):163–169.
75. Pan, Z., Yang, Y., Li, X., and Xu, S.-J. (2020). The complexity of total edge domination and some related results on trees. *Journal of Combinatorial Optimization*, 40:571–589.
76. Panda, B. and Das, S. K. (2003). A linear time recognition algorithm for proper interval graphs. *Information Processing Letters*, 87(3):153–161.
77. Paspasan, M. N. S. and Canoy, S. (2016). Edge domination and total edge domination in the join of graphs. *Applied Mathematical Sciences*, 10(22):1077–1086.
78. Paul, S. and Ranjan, K. (2022). Results on vertex-edge and independent vertex-edge domination. *Journal of Combinatorial Optimization*, 44(1):303–330.

79. Peeters, R. (2003). The maximum edge biclique problem is np-complete. *Discrete Applied Mathematics*, 131(3):651–654.
80. Prieto, E. and Sloper, C. (2003). Either/or: Using vertex cover structure in designing fpt-algorithms—the case of k-internal spanning tree. In *Workshop on Algorithms and Data Structures*, pages 474–483. Springer.
81. Salamon, G. (2009). Approximating the maximum internal spanning tree problem. *Theoretical Computer Science*, 410(50):5273–5284.
82. Salamon, G. (2010). A survey on algorithms for the maximum internal spanning tree and related problems. *Electronic Notes in Discrete Mathematics*, 36:1209–1216.
83. Salamon, G. and Wiener, G. (2008). On finding spanning trees with few leaves. *Information Processing Letters*, 105(5):164–169.
84. Seinsche, D. (1974). On a property of the class of n-colorable graphs. *Journal of Combinatorial Theory, Series B*, 16(2):191–193.
85. Senthilkumar, B., Naresh Kumar, H., and Venkatakrishnan, Y. B. (2021). Bounds on total edge domination number of a tree. *Discrete Mathematics, Algorithms and Applications*, 13(02):2150011.
86. Spinrad, J., Brandstädt, A., and Stewart, L. (1987). Bipartite permutation graphs. *Discrete Applied Mathematics*, 18(3):279–292.
87. Srikant, R., Sundaram, R., Singh, K. S., and Rangan, C. P. (1993). Optimal path cover problem on block graphs and bipartite permutation graphs. *Theoretical Computer Science*, 115(2):351–357.
88. Tan, J. (2008). Inapproximability of maximum weighted edge biclique and its applications. In *International Conference on Theory and Applications of Models of Computation*, pages 282–293. Springer.
89. Velammal, S. (2014). Equality of connected edge domination and total edge domination in graphs. *International Journal of Enhanced Research in Science Technology and Engineering*, 3:198–201.

90. Williamson, D. P. and Shmoys, D. B. (2011). *The design of approximation algorithms*. Cambridge university press.
91. Yannakakis, M. and Gavril, F. (1980). Edge dominating sets in graphs. *SIAM Journal on Applied Mathematics*, 38:364–372.
92. Zhao, Y., Liao, Z., and Miao, L. (2014). On the algorithmic complexity of edge total domination. *Theoretical Computer Science*, 557:28–33.

List of Publications Out of Thesis

- B. Brešar, A. Pandey, and G. Sharma, *Computational aspects of some vertex sequences of Grundy domination-type*, **Indian Journal of Discrete Mathematics**, 8(1), 21–38, 2022. (Presented results in **ICGT 2022**, The 11th International Colloquium on Graph Theory and combinatorics held at the University of Montpellier, France from 4-8th July 2022.)
- G. Sharma, A. Pandey, and M. C. Wigal, *Algorithms for maximum internal spanning tree problem for some graph classes*, **Journal of Combinatorial Optimization**, 44(2), 1–27, 2022.
- M. A. Henning, A. Pandey, G. Sharma, and V. Tripathi, *Algorithms and Hardness Results for Edge Total Domination Problem in Graphs*, **Theoretical Computer Science**, 982, 114270, 2023.
- B. Brešar, A. Pandey, and G. Sharma, *Computation of Grundy dominating sequences in (co-)bipartite graphs*, **Computational and Applied Mathematics**, 42(8), 359, 2023.
- A. Pandey, G. Sharma, and N. Jain, *Maximum weighted edge biclique problem on bipartite graphs*, in proceedings of **CALDAM 2020**, **Lecture Notes in Computer Science**, Vol. 12016, pp. 116-128, Springer.
- G. Sharma, and A. Pandey, *Computational Aspects of Double Dominating Sequences in Graphs*, in proceedings of **CALDAM 2023**, **Lecture Notes in Computer Science**, Vol. 13947, pp. 284-296, Springer. (Journal version is under review in **Discussiones Mathematicae Graph Theory**).
- G. Sharma, A. Pandey, and R. Kamra, *Neighbor-Locating Colorings: Hardness, Algorithms, and Bounds*, presented in **ICRAGAA 2023**, International Conference on Recent Advances in Graph Theory and Allied Areas held at St. Aloysius College, Thrissur, Kerala, India from 2-4th February 2022 (Conference proceedings will be published in **AKCE International Journal of Graphs and Combinatorics**).

CURRICULUM VITAE

Name : Gopika Sharma
Date of birth : July 21, 1992
Nationality : Indian
Affiliation : IIT Ropar
Email : *2017maz0007@iitrpr.ac.in*
Other Email : *gopikasharma09@gmail.com*

Academic Qualifications

- PhD: Pursuing from Department of Mathematics, IIT Ropar (January 2018 - March 2024).
- M.Sc: IIT Guwahati (2014).
- B.Sc. (Mathematics): Banasthali University (2012).

Award and Honours

- CSIR NET 2016, All India Rank - 48
- GATE-2017, All India Rank - 271.

Presentations and Participation in Workshops and Conferences

- Presented a paper in “**9th Annual International Conference on Algorithms and Discrete Applied Mathematics (CALDAM 2023)**” held at DA-IICT Gandhinagar during 09th February to 11th February, 2023.
- Participated in “**Indo-Dutch Pre-Conference School on Algorithms and Combinatorics**” held at DA-IICT Gandhinagar during 06-07th February, 2023.

- Presented a paper in **“International Conference on Recent Advances in Graph Theory and Allied Areas (ICRAGAA 2023)”** held at St. Aloysius College, Elthuruth, Thrissur during 02th February to 04th February, 2023.
- Presented a contributed talk in the **“Annual Research Day of Department of Mathematics, Cynosure-2022”** held on December 10, 2022 at IIT Ropar.
- Presented a paper in **“The 11th International Colloquium on Graph Theory and Combinatorics (ICGT 2022)”** held at University of Montpellier, France during 4-8th July, 2022.
- Presented a contributed talk in the **“Online International Workshop on Domination in Graphs (IWDG-2021)”** held during November 14-16, 2021 jointly organized by IIT Ropar and Academy of Discrete Mathematics and Applications.
- Presented a contributed talk in the **“Annual Research Day of Department of Mathematics, Cynosure-2021”** held on December 21, 2021 at IIT Ropar.
- Participated in **“Workshop on Parameterized Complexity”** held at IISER Pune, India during 6-8th March, 2020.
- Participated in **“6th Annual International Conference on Algorithms and Discrete Applied Mathematics (CALDAM 2020)”** held at IIT Hyderabad during 13th February to 15th February, 2020.
- Presented a contributed talk in **“Indo-French Pre-Conference School on Algorithms and Combinatorics”** held at IIT Hyderabad during 10-11th February, 2020.
- Participated in **“Workshop on Introduction to Spectral Graph Theory”** held at IIT Ropar during 16-20th November, 2019.
- Participated in **“Workshop on Probabilistic Methods in Graph Theory and Combinatorics”** held at IIT Ropar during 7-10th November, 2019.
- Attended **“ACM-India Summer School on Graph Theory and Graph Algorithms”** held at PSG College of Technology, Coimbatore during 21 May-8th June, 2018.