Variants of Domination and Their Algorithmic Complexities

Thesis submitted to the
Indian Institute of Technology Guwahati
for the award of the degree

of

Doctor of Philosophy

Submitted by
Sasmita Rout
186123015

Under the guidance of
Prof. Gautam Kumar Das

Department of Mathematics

Indian Institute of Technology Guwahati
Guwahati-781039, Assam, India

February, 2025

Declaration

I certify that:

a. The work contained in this thesis is original and has been done by me under the guidance of my supervisor.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. I have followed the guidelines provided by the Institute in preparing the thesis.

d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

Sasmita Rout
186123015

Copyright

Sasmita Rout
186123015

Certificate

This is to certify that this thesis entitled, **"Variants of Domination and Their Algorithmic Complexities "**, being submitted by **Sasmita Rout**, to the Department of Mathematics, Indian Institute of Technology Guwahati, for partial fulfillment of the award of the degree of Doctor of Philosophy, is a bonafide work carried out by her under my supervision and guidance. The thesis, in my opinion, is worthy of consideration for award of the degree of Doctor of Philosophy in accordance with the regulation of the institute. To the best of my knowledge, it has not been submitted elsewhere for the award of the degree.

Prof. Gautam K. Das
Professor
Dept. of Mathematics
IIT Guwahati

Dedicated to

my husband and mother

whose love and support paved my path of success

Acknowledgements

My six-year Ph.D. journey has been a roller coaster of ups and downs. This journey would not have been completed without the emotional and mental support of several people, and my thesis work would be incomplete without acknowledging their contributions.

First and foremost, I express my deepest gratitude to my thesis supervisor, Prof. Gautam Kumar Das, for allowing me to carry out my research under his supervision and for introducing me to the fascinating field of approximation algorithms. I vividly remember the times when I gave stupid answers to his questions and the moments when I finally understood some of the points he raised, even if it took me months to understand them. I also want to thank my supervisor for granting me flexibility with my schedule, considering my circumstances. His guidance and continuous support have been invaluable throughout my journey at IIT Guwahati. I am deeply grateful for his assistance, both academically and personally. I could not have asked for a better advisor for my PhD.

Besides my advisor, I would also like to thank my doctoral committee, Dr. Pinaki Mitra, Dr. Deepanjan Kesh, and Prof. Ashok Singh Sairam, for their insightful and valuable comments which encouraged me to widen my research from various perspectives. I am grateful to the anonymous reviewers of my papers for their insightful suggestions and feedback. My sincere thanks to the Ministry of Human Resource Development, Government of India for providing me with financial support for pursuing PhD at IIT Guwahati. I want to express my heartfelt gratitude to the director, the deans, and the entire management team of IIT Guwahati for their collective efforts in making this institute a hub for world-class education and research. I also thank the Head of the Department of Mathematics, IIT Guwahati, and other teaching and non-teaching staff members of the Department for their support in administrative and technical works.

I would also like to extend my gratitude to Dr. Pawan Kumar Mishra for his collaborative work and for co-authoring a publication. I am deeply thankful to my senior, Dr. Sangram Kishore Jena, for his invaluable guidance and support.

I want to express my heartfelt gratitude to some very special people in my life for their unwavering support. Tina, Munu, Amiya, Sachin, Soma, Chandan, Preeti, Lalitha, and Swagat, thank you for always being there for me. I thank my dearest friend Nibedita for enduring my erratic emotions during my low points and encouraging me to fight back. I would also like to thank all my teachers, from my early education to my postgraduate studies, who have helped me learn and grow. I am incredibly grateful to Mr. Sanjay Kumar Nanda, my class 10^{th} teacher, and Prof. Tripti Swarnkar, my postgraduate teacher, for their belief in me.

Most importantly, none of this would have been possible without the love and patience of my family. I am grateful to my father, mother and brother for their unconditional love and consistent support in every event of my life. A special thanks to my mother who always stood behind me to support me even after my marriage. I would also like to thank my in-laws. I am forever indebted to my husband, Dr. Debabrata Senapati. His love, constant encouragement, and unwavering faith in me have made all of this possible. Through all the ups and downs over the past few years, he has been incredibly supportive. I cannot thank him enough for being such an excellent life partner and friend. Finally, my heartfelt thanks to my daughter, Purvi. Our mother-daughter relationship is a treasure beyond measure, and her presence in my life has been the ultimate source of inspiration and joy. I thank her for being understanding and cooperative when I could not spend quality time with her.

Last but not least, I want to express my gratitude to the Supreme Soul for providing me with more than I could have ever imagined. Before concluding, once again, thanks to you all.

Sasmita Rout

Abstract

The Dominating Set Problem (DSP) is a classical combinatorial optimization problem that aims to find the smallest dominating set (DS) in a given graph $G = (V, E)$. Due to its numerous real-world applications across various domains, such as facility location, social networks, network design, security, and resource allocation, the dominating set and its variants have been extensively studied. In this thesis, we explore several variants of the dominating set, specifically the semi-total dominating set, total dominating set, and total Roman dominating set. All the problems listed here are NP-hard, and none of them admit a constant factor approximation algorithm for general graphs unless $P = NP$. This challenge motivated us to study these problems within special graph classes. Unit disk graphs (UDGs) are geometric intersection graphs, where nodes correspond to disks, and there is an edge between two nodes if their corresponding disks intersect. Exploring domination and its variants within unit disk graphs (UDGs) is particularly significant due to their pivotal role as models for wireless communication networks, mobile Ad-Hoc networks (MANETs), and sensor networks. Given the relation of UDGs with domination theory, we investigate most of these problems in UDGs and a few of them in grid graphs (GGs), and trees.

Firstly, we study the semi-total dominating set (T2DS) and we are the first to introduce the semi-total dominating set problem in unit disk graphs. We establish that the T2DS problem is NP-complete in UDGs and propose a 6-factor approximation algorithm. The running time of the algorithm is $O(n \log k)$, where k is the size of the maximal independent set. We also propose a $2 + \ln(\Delta + 1)$-factor approximation algorithm for the semi-total domination problem in general graphs, where Δ is the degree of the graph. Our proposed approximation algorithm is an improvement over the approximation factor $2 + 3\ln(\Delta + 1)$ given in the literature. We have also observed that the problem admits a PTAS with approximation factor $(2 + \epsilon)$.

Secondly, we study the total dominating set (TDS) problem and propose a 7.79-factor approximation algorithm and a 6.29-factor approximation algorithm for the TDS problem in UDGs, with running time $O(n^2)$ and $O(n^2 m)$, respectively, where n is the number of vertices and m is the number of edges of the graph. We have also shown that the TDS problem is NP-complete when restricted to grid graphs, a subclass of UDGs.

Next, we study the total Roman dominating set (TRDS), a well-known variant of Roman domination. We establish that the TRDS problem is NP-complete in UDGs and propose a 10.5-factor approximation algorithm and a 6.15-factor approximation algorithm for the same with time complexities $O(n \log k)$ and $O(n^2 m)$, respectively, where n is the number of vertices, m is the number of edges and k is the size of the maximal independent set of the graph.

Finally, we delved into the algorithmic aspect of the TRDS problem in trees and introduced a dynamic programming-based algorithm to yield an optimal solution. Our algorithm operates in linear time, offering an efficient approach to tackle the problem in tree structures.

Contents

List of Figures

List of Algorithms

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

List of Abbreviations

APX	The class of polynomial-time constant approximable problems
DS	Dominating Set
DSP	Dominating Set Problem
DTIME	Deterministic Time
GGs	Grid Graphs
IS	Independent Set
MANET	Mobile Ad-hoc Network
MDS	Minimum Dominating Set
MDSP	Minimum Dominating Set Problem
MIS	Maximum Independent Set
MSC	Minimum Set Cover
MTDS	Minimum Total Dominating Set
MT2DS	Minimum Semi-total Dominating Set
MRDS	Minimum Roman Dominating Set
NP	The class of polynomial-time varifiable problems
NP-complete	The class of problems that are NP and NP-hard
NP-hard	The class of non-deterministic polynomial-time hard problems
P	The class of deterministic polynomial-time solvable problems
PTAS	Polynomial Time Approximation Scheme

RDF	Roman Dominating Function
RDS	Roman Dominating Set
SC	Set Cover
SCP	Set Cover Problem
TDS	Total Dominating Set
TRDF	Total Roman dominating Function
TRDS	Total Roman Dominating Set
UDG	Unit Disk Graph
VC	Vertex Cover
VCP	Vertex Cover Problem
VLSI	Very Large Scale Integration

List of Symbols

Symbol	Meaning
$\mathbb{R}^2$	$\{(a, b) \mid a, b \in \mathbb{R}\}$
$\mathcal{P}$	A set of points in $\mathbb{R}^2$
$p \in \mathcal{P}$	p is a member of $\mathcal{P}$
$\delta(p, q)$	The Euclidean distance between p and q
$d(p, q)$	The number of edges on a shortest path between two vertices p and q
Δ	The degree of the graph G
$U(p)$	The unit disk centered at point p
$U(P)$	The set of unit disks centered at the points in P
$\mathcal{R}$	A rectangular region in the plane
:	Such that
$\mathcal{Q} \subseteq \mathcal{P}$	$\mathcal{Q}$ is a subset of $\mathcal{P}$
$\mathcal{P} \cap \mathcal{Q}$	The intersection of $\mathcal{P}$ and $\mathcal{Q}$
$\mathcal{P} \cup \mathcal{Q}$	The union of $\mathcal{P}$ and $\mathcal{Q}$
$\mathcal{P} \setminus \mathcal{Q}$	The set minus of $\mathcal{P}$ and $\mathcal{Q}$
$\lvert \cdot \rvert$	The cardinality of a set
$\sum$	The addition of a sequence of numbers
$\square$	The end of a proof
$\emptyset$	The empty set
$\{.\}$	The set notation
$G = (V, E)$	An undirected connected simple graph with vertex set V and edge set E

Symbol	Meaning
$e \in E$	e is a member of E
n	The cardinality of V (or) the cardinality of a point set $\mathcal{P}$
m	The cardinality of E
uv	An edge with end vertices u and v
$N_G(.)$	The open neighborhood of a vertex
$N_G[.]$	The closed neighborhood of a vertex
K_n	The complete graph of n vertices
ϵ	A positive real number less than 1
$O(\cdot)$	The asymptotic big-oh notation
$G \square H$	The cartesian product of the graphs G and H
$f : V \longrightarrow A$	f is a function from V onto A
$f(x) = c$	The value associated with the vertex x is c
$W(f)$	Weight associated with the function f
V_j	The set of vertices with Roman values j
$\gamma(G)$	The domination number of G
$\gamma_t(G)$	The total domination number of G
$\gamma_{t2}(G)$	The semi-total domination number of G
$\gamma_R(G)$	The Roman domination number of G
$\gamma_{tR}(G)$	The total Roman domination number of G
$\Gamma_t(G)$	The upper total domination number of G
$\Gamma_{pr}(G)$	The upper paired domination number of G
$P_2(G)$	The 2-*packing* number of G
$\alpha'(G)$	The matching number of G
$x \overset{G}{\rightsquigarrow} y$	Denote a path from x to y consisting of multiple edges
$x \overset{G}{\rightsquigarrow} y \overset{G}{\rightsquigarrow} z$	Denote a cycle consisting of paths from x to y and then y to z
$H(m)$	m-*th* Harmonic number
$< \mathbb{U}, \mathbb{S} >$	Set cover instance
$\mathbb{U}$	Universal set
$\mathbb{S}$	Set of subsets of $\mathbb{U}$
C^*	An optimal set cover

Symbol	Meaning
D^*	An optimal dominating set
f^*	An optimal TRDF
$T(x)$	A tree T_x rooted at x
I	Maximal independent set of G
$C(u)$	Set of children of a vertex in the tree T

LIST OF ALGORITHMS

Chapter 1

Introduction

The *dominating set problem* (DSP) is a classical combinatorial optimization problem that aims to identify the smallest *dominating set* (DS) in a simple undirected graph $G = (V, E)$. In G, a *dominating set* $D \subseteq V$ is defined as a subset of vertices such that every vertex in V is either included in D or is adjacent to at least one vertex in D. It can also be expressed using a function f as $D = \{v \in V : f(v) = 1\}$, where $f : V \to \{0, 1\}$ and each vertex $u \in V$ with $f(u) = 0$ is adjacent to at least one vertex $v \in V$ with $f(v) = 1$. The concept of domination was first introduced by C. Berge in the year 1958 [12]. A graph can have multiple DS (refer to Figure 1.1, where the set of red vertices in each subfigure represents a *dominating set* of the given graph G). However, a minimum-size DS is called a *minimum dominating set* (MDS), and the MDS cardinality is called the *domination number*. It is denoted by $\gamma(G)$. For a vertex $v \in V$, we denote $N_G(v)$ and $N_G[v]$ as the open and closed neighborhood of v, respectively, and define them as $N_G(v) = \{u : uv \in E\}$ and $N_G[v] = N_G(v) \cup \{v\}$. A vertex $v \in D$ dominates itself and its neighbors (i.e., $N_G[v]$), where v is the dominator and the vertices in $N_G(v)$ are the dominatees. In computational complexity theory, the *dominating set problem* is a classical NP-complete problem [49]. Along with the domination problem, its variants are also generally hard in general graphs. Keeping an eye on the theoretical importance, practical relevance, and rich structure of *dominating set* and its variants, the researcher analyzed these problems in various graph classes such as *unit disk graphs* (UDGs), planar graphs, trees, etc., and explored them in terms of both theoretical and practical dimensions.

1. INTRODUCTION

The literature on domination and its variants can be found in [51, 53, 57, 59, 71, 95]. The domination and its variants have countless real-world applications across various domains. Some important applications are facility location, social network, network design, security, resource allocation, etc. [4, 46, 51]. Out of many such applications, some of them are briefly discussed below.

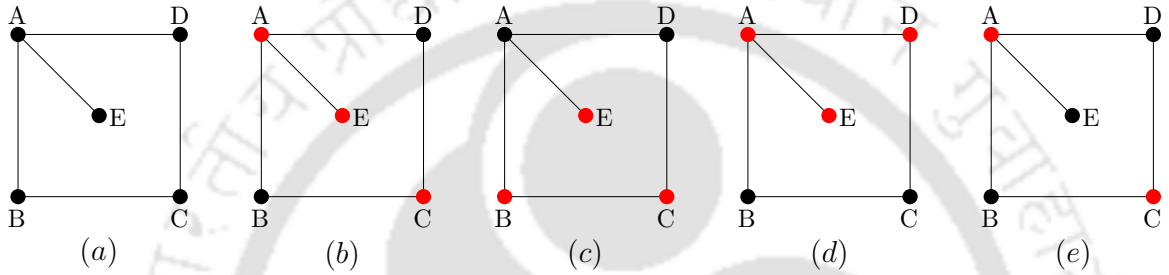

Figure 1.1: *(a) Graph G, (b) DS of G, (c) DS of G, (d) DS of G, and (e) MDS of G*

In computer networks, dominating sets are used to design efficient routing protocols and network management algorithms, which in turn help to identify the critical nodes that can regulate or influence the behavior of the entire network. In the design of wireless sensor networks, dominating sets are used to determine the optimal placement of sensor nodes so that the entire network is efficiently monitored. It also reduces redundancy and conserves energy, as only a few nodes need to be active at any given time. It is also used for floor planning and layout optimization in the *very large scale integration* (VLSI) design to ensure a reduction in the overall area and power consumption of the integrated circuit. The facility location problem and *dominating set* are related in the context of optimization and decision making in network design and infrastructure planning. The purpose of the facility location problem is to determine the optimal locations for opening facilities such as hospitals, warehouses, fire stations, distribution centers, or service centers to serve a set of customers or clients while minimizing overall cost. In many cases, facilities need to be strategically located so that they can effectively cover and serve a certain geographic area or population.

Looking into the varied applications, the concept of domination has been the subject of

extensive research since the 1950s, as evidenced by numerous studies documented in references [39, 45, 51, 52, 57, 106].

Researchers have explored various forms of domination in graphs to meet specific requirements and demands, incorporating additional constraints relevant to different fields. Some notable variations include total domination, Roman domination, Italian domination, perfect Roman domination, and perfect Italian domination. [10, 19, 27, 48, 58, 62, 63]. A few of these are listed below briefly, with their applications.

Given a graph G without any isolated vertex, a subset $D_t \subseteq V(G)$ is said to be a *total dominating set* (TDS) of G if (i) D_t is a *dominating set* of G, and (ii) the vertices in D_t induce a subgraph with no isolated vertex. We refer to the first property of total domination as *domination property* and the second one as *total property*. The *total dominating set* with minimum cardinality is called *minimum total dominating set* (MTDS), and the size of the *minimum total dominating set* is called *total domination number*, $\gamma_t(G)$ (Figure 1.2 illustrates the total dominating sets of the graph G). It is important to note that total dominating sets are not defined for graphs containing isolated vertices. The *total dominating*

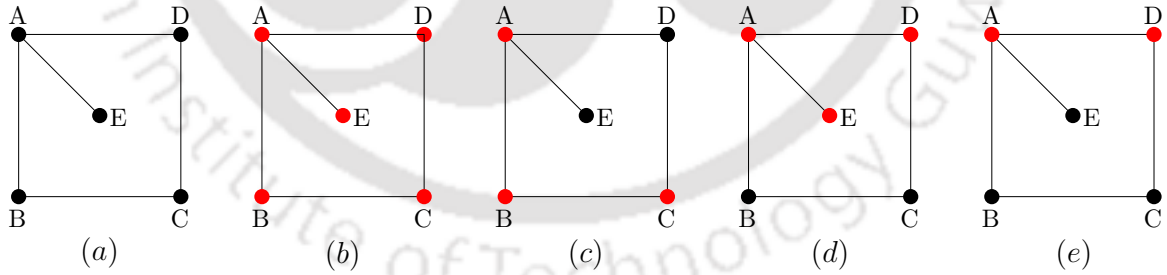

Figure 1.2: *(a) Graph G, (b) TDS of G, (c) TDS of G, (d) TDS of G, and (e) MTDS of G*

set properties are useful in networks where every site (including the monitoring devices) is adjacent to at least one monitoring device. This property enhances the fault tolerance and reliability of the network [72]. A TDS also helps in planning multiple robot paths to ensure complete area coverage in robotics.

A subset $D_{t2} \subseteq V$ is said to be a *semi-total dominating set* (T2DS) of G if (i) D_{t2} is a *dominating set*, and (ii) for each $u \in D_{t2}$, there exists a vertex $v \in D_{t2}$ such that $d(u, v) \leq 2$, where $d(.,.)$ represents the minimum number of edges connecting two vertices in G. We refer to the first property of the $T2DS$ as *domination property* and the second one as *semi-total property*. The *semi-total dominating set* with minimum cardinality is the *minimum semi-total dominating set* (MT2DS), and the corresponding cardinality is the *semi-total domination number*. We denote the *semi-total domination number* as $\gamma_{t2}(G)$. Like TDS, T2DS is also not defined for graphs containing isolated vertices (Figure 1.3 illustrates a few semi-total dominating sets of the graph G). Among the various applications of the T2DS

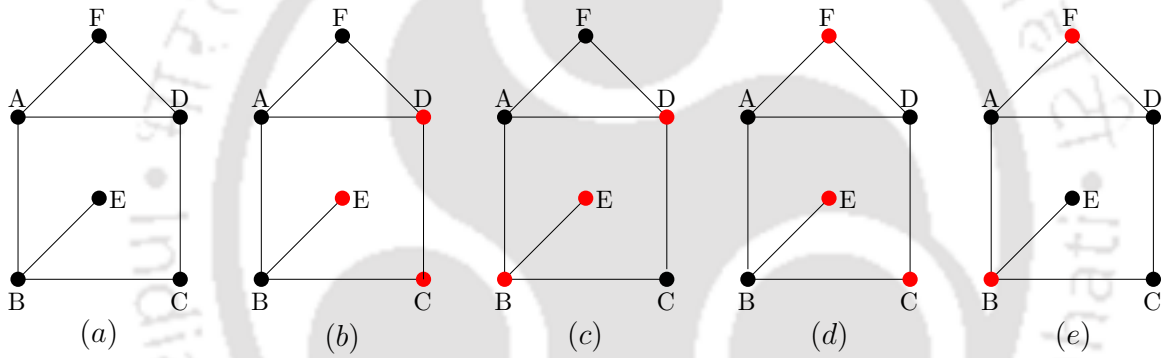

Figure 1.3: *(a) Graph G, (b) T2DS of G, (c) T2DS of G, (d) T2DS of G, and (e) MT2DS of G*

problem, several notable ones are listed below. T2DS problems are pivotal in designing robust and fault-tolerant networks. The *semi-total property* of T2DS ensures that each node in the *dominating set* has a neighbor, which is crucial for maintaining communication paths even if some nodes fail. This property also facilitates efficient broadcasting within the network, as nodes can receive broadcasted messages directly or through a neighbor, ensuring message propagation throughout the entire network. In addition to network design, semi-total domination is instrumental in studying biological networks and robotics. The interactions among key genes and the connectivity among leading robots can be effectively analyzed using the *semi-total property* of the *semi-total dominating set*. This analysis helps to understand and enhance the robustness and coordination within these systems. Some of the applications can be found in [42].

Among the various forms of domination in graphs, one variant called Roman domination has recently gained significant popularity. The *Roman dominating set* (RDS) is an ordered partition of V, say (V_0, V_1, V_2) induced by a function $f : V \to \{0, 1, 2\}$ called *Roman dominating function* (RDF) such that (i) $V_j = \{v \in V : f(v) = j\}$, for $j = 0, 1, 2$, and (ii) for each $v \in V_0$, there exists at least a vertex $u \in V_2$ such that $uv \in E$. The RDF with minimum weight, $f(V) = \sum_{u \in V} f(u)$, is called the *Roman domination number*, and it is denoted by $\gamma_R(G)$ (Figure 1.4 illustrates a few Roman dominating sets of the graph G). Given a graph

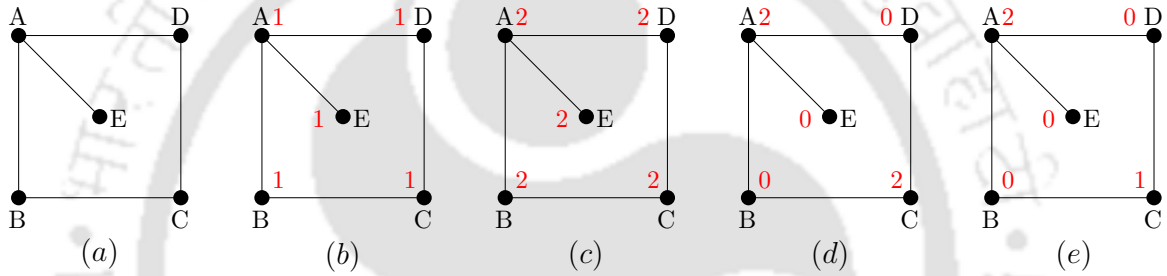

Figure 1.4: *(a) Graph G, (b) RDS of G, (c) RDS of G, (d) RDS of G, and (e) Minimum RDS of G*

G, the *Roman domination problem* (RDP) aims to find a *Roman dominating function* of G of minimum weight. The problem holds both historical and mathematical interests. In the fourth century A.D., the Roman Emperor Constantine the Great employed a similar strategy for defending the Roman Empire with limited resources. He ordered each region without an army platoon to have at least one neighboring region with two army platoons. This strategy ensured that during a crisis, an unsecured region (with no platoons) could be protected by transferring a platoon from a neighboring secured region (with two platoons) without compromising the defense of the secured region. Mathematically, it can be viewed as a 3-coloring graph problem (with colors 0, 1, and 2) with the additional constraint that any vertex with color 0 is adjacent to at least one vertex with color 2.

Technically, it was first introduced by Cockayne et al. [28] and was motivated by an article [100], which was based on legion deployment for better security with limited resources.

RDP is NP-complete for general graphs [38]. It is also NP-complete when restricted to bipartite graphs, split graphs, and planar graphs [28]. C. Padamutham et al. gave linear time algorithms for the *Roman domination problem* in bounded treewidth graphs, chain graphs, and threshold graphs [92]. The RDP is linear-time solvable in interval graphs and co-graphs [79]. In [79], authors also gave polynomial-time algorithms for D-octopus graphs and AT-free graphs. The RDP is also studied on circulant graphs, generalized Peterson graphs, and Cartesian product graphs [107]. Some other results and variations of Roman domination can be found in [1, 20, 23, 40, 80, 99, 107]. One of the well-known variants of Roman domination, namely total Roman domination, is discussed below briefly.

The *total Roman dominating set* (TRDS) is an ordered partition of V, say (V_0, V_1, V_2) induced by a function, $f : V \to \{0, 1, 2\}$ called total Roman dominating function (TRDF) such that (i) f is a *Roman dominating function* (*Roman property*), and (ii) the induced subgraph $G[V_1 \cup V_2]$ does not contain any isolated vertex (*total property*). The TRDF of minimum weight, $f(V) = \sum_{u \in V(G)} f(u)$, is called the *total Roman domination number* (Figure 1.5 illustrates a few total Roman dominating sets of the graph G). We define *total Roman domination number* $\gamma_{tR}(G)$ as the minimum weight among all TRDFs on G. Like TDS and T2DS, TRDS is also not defined for graphs containing isolated vertices. The TRDS problem has several practical applications across various fields, including network security, social networks, telecommunications, resource allocations, etc. Some references that discuss the TRDS problem's applications in various fields are [35, 41, 88].

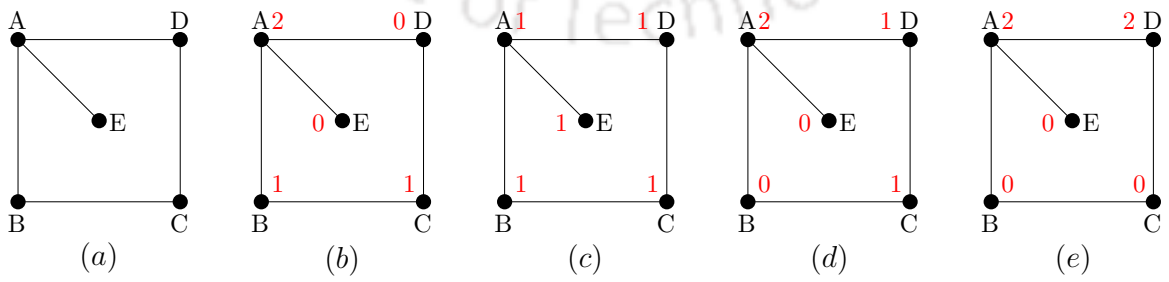

Figure 1.5: *(a) Graph G, (b) TRDS of G, (c) TRDS of G, (d) TRDS of G, and (e) Minimum TRDS of G*

Undecidable problems are the decision problems for which algorithms do not exist. In contrast, *decidable problems* are the decision problems for which algorithms exist, where each algorithm provides a correct yes or no answer for every input instance in a finite amount of time. P is the class of all decision problems solvable in polynomial time. NP is the class of problems for which a polynomial verification algorithm exists for the proposed solution. NP - *complete* problems are the most complex problems in NP. These problems are in NP and as hard as any problem in NP. Unlike NP - *complete* problems, NP - *hard* problems do not have to be present in NP, i.e., these problems may or may not be verifiable in polynomial time.

Approximation algorithms are used for NP - *hard* problems and aim to find solutions close to the optimal solution in polynomial time. An algorithm for an optimization problem (whether minimization or maximization) is termed a *ρ-factor approximation algorithm* if, for every instance of the problem, it produces a feasible solution within a factor ρ of the optimal solution. It operates in polynomial time relative to the input size. Here, ρ is the algorithm's approximation ratio or approximation factor.

- Maximization problem: $\rho = \frac{|C^*|}{|C|}$, where C^* is the optimal solution, C is the approximate solution and $0 < |C| < |C^*|$.
- Minimization problem: $\rho = \frac{|C|}{|C^*|}$, where C^* is the optimal solution, C is the approximate solution and $0 < |C^*| < |C|$.

A *polynomial-time approximation scheme* (PTAS) for an optimization problem is a collection of algorithms that provides a way to get arbitrarily close to the optimal solution within polynomial time. It takes an additional parameter $\epsilon > 0$ and yields a $(1 + \epsilon)$-factor approximation algorithm. The running time is polynomial in the size of the input instance of the problem and ϵ.

An *independent set* is a set of vertices in which no two vertices in the set are adjacent. A *maximal independent set* of a graph $G = (V, E)$ is a set of vertices $V' \subseteq V$ such that no vertices can be added to V' without creating an edge within the set. The largest maximal independent set is called the *maximum independent set*. It is important to note that a maximal independent set may not be a maximum independent set. An *independent dominating set* is a *dominating set*, which is also an *independent set*.

An *intersection graph* of objects is a graph, where the vertex set is the set of objects and the edges connect objects if objects are within a specific distance. A *Unit Disk Graph* (UDG) is a special type of intersection graph where the objects are unit disks. In this graph, the centers of the disks represent the nodes, and an edge exists between two nodes if the Euclidean distance between their centers is at most 1. For an illustration see Figure 1.6. Exploring domination and its variants within *unit disk graphs* (UDGs) is particularly significant due to their pivotal role as models for wireless communication networks, mobile Ad-Hoc networks (MANETs), and sensor networks.

Given the importance of domination theory in modeling wireless communication networks, mobile Ad-Hoc networks, and sensor networks using UDGs, our research aims to investigate some variants of domination, focusing specifically on semi-total domination, total domination and total Roman domination within designated graph classes, particularly in UDGs.

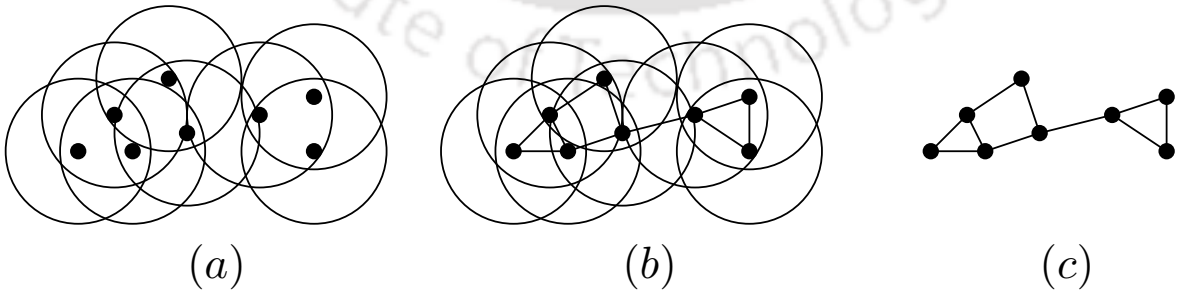

Figure 1.6: *(a) A set of unit disks on an Euclidean plane, (b) The UDG correspond to the set of disks, and (c) UDG*

1.1 Scope of the Thesis

The optimization problems such as the *semi-total dominating set* (T2DS) problem, *total dominating set* (TDS) problem, and *total Roman dominating set* (TRDS) problem are NP-hard, and none of them admits constant factor approximation algorithm for general graphs, unless $P = NP$. This motivated many researchers to investigate the problems for other graph classes. Looking into the importance of UDGs in modeling wireless communication networks, mobile Ad-Hoc networks, and sensor networks, we study all these problems, specifically in unit disk graphs, and show that these problems are also NP-hard in UDGs. Next, we propose constant factor approximation algorithms for these problems. We have also studied the total dominating set problem in grid graphs and shown that it is NP-complete when the graph is restricted to grid graphs, a subclass of UDGs. We propose a linear time algorithm to get an optimal solution for the total Roman domination problem in trees.

1.2 Organization of the Thesis

The thesis contains seven chapters and is organized as follows.

Chapter 2. Literature Review: In this chapter, we detail the existing literature, outlining key works and their significant findings.

Chapter 3. Semi-total Domination in UDGs and General Graphs: In this chapter, we discuss the *semi-total dominating set*. Here, we have shown that the problem is NP-complete in UDGs and proposed a 6-factor approximation algorithm. In addition to this, we have also given a $2+\ln(\Delta+1)$-factor approximation algorithm for general graphs, where Δ is the degree of the graph.

Chapter 4. Total Domination in UDGs: In this chapter, we study the *total dominating set* problem in *unit disk graphs* and propose two approximation algorithms with approximation factors 7.79 and 6.29, which are the improvements over the best known 8-factor approximation algorithm. The running times of the algorithms are

$O(n^2)$ and $O(n^2m)$, respectively, where n is the number of vertices and m is the number of edges in the graph. In this chapter, we also strengthened the complexity of the problem by demonstrating its NP-completeness, even in the case of grid graphs.

Chapter 5. Total Roman Domination in UDGs: In this chapter, we focus on the *total Roman dominating set* problem in UDGs and show that the problem is NP-complete. We propose a 10.5-factor and a 6.15-factor approximation algorithm for the same with time complexities $O(n \log k)$ and $O(n^2m)$, respectively, where n is the number vertices, m is the number of edges, and k is the size of the maximal independent set of the graph.

Chapter 6. Total Roman Domination in Trees: In this chapter, we study the *total Roman dominating set* problem in trees and propose an optimal solution using dynamic programming, demonstrating its efficiency with a linear time complexity.

Chapter 7. Conclusion and Future Work: Finally, in Chapter 7, we discuss the concluding remarks of the thesis and some open problems for future research endeavors.

Chapter 2

Literature Review

In this chapter, we begin by reviewing key literature on *dominating set* (DS), followed by a discussion of the state-of-the-art results for the problems under consideration: the *total dominating set* (TDS), *semi-total dominating set* (T2DS), and *total Roman dominating set* (TRDS). The chapter is divided into three sections. In each section, we first define the specific problem and then present the existing results.

Dominating Set: *Let $G = (V, E)$ be a simple undirected graph. A subset $D \subseteq V$ is said to be a dominating set (DS) of G if each vertex in V is either present in D or is adjacent to at least one vertex in D. The dominating set of minimum size is called a minimum dominating set (MDS). The cardinality of a minimum dominating set is called the domination number and is denoted by $\gamma(G)$. Given graph G, the dominating set problem aims to find a dominating set of minimum size.*

The concept was first introduced in the book "*Theory of graphs and its applications*" by C. Berge in the year 1958 [13]. The MDS problem is a classical NP-complete problem. This means that there is no known polynomial-time algorithm to solve the MDS problem unless P = NP. Since the problem is NP-complete, most of the solutions are focused on approximation algorithm. It is important to note that all approximation and inapproximability results for the *set cover problem* also apply to the *dominating set problem*. This is

because a pair of polynomial-time L-reductions exists between the minimum *dominating set problem* and the *set cover problem* [73]. These reductions imply that an efficient algorithm for the *set cover problem* would yield an efficient algorithm for the MDS problem and vice versa. Furthermore, these reductions preserve the approximation ratio: a polynomial-time ρ-approximation algorithm for *set cover problem* would provide a polynomial-time ρ-approximation algorithm for the MDS problem, and vice versa. A simple greedy algorithm can be used to get a $(\ln n + 1)$-factor approximation algorithm for the MDS problem [25].

The MDS problem is also NP-hard in many sub-classes of graphs such as: bipartite graph [78], perfect elimination bipartite graphs [16], UDGs [26] etc. In [93], authors showed that for any $\epsilon > 0$, the MDS problem does not admit a $(1 - \epsilon) \ln n$ approximation algorithm for star-convex bipartite graphs [1] with n vertices unless $NP \subseteq DTIME(n^{O(\log \log n)})$. They also proved that the MDS problem is linear-time solvable for bounded degree star-convex bipartite graphs. Additionally, they presented polynomial time algorithms to solve the MDS problem in circular-convex bipartite graphs [2] and triad-convex bipartite graphs [3]. P. Damaschke et al. gave a polynomial time algorithm to find the MDS in a convex bipartite graph [32] [4]. In 1993, J Mark Keil proved that the MDS problem is NP-complete in circle graphs [5] [75]. In [33], authors gave a 8-factor approximation algorithm and a $(2 + \epsilon)$-factor approximation scheme for the MDS problem in circle graphs. In [82] Marathe et al. gave a 5-factor approximation algorithm for the MDS problem in UDG. In 2006, Neiberg and Hurink presented a PTAS for the MDS problem in UDGs. The running time of the PTAS is $n^{O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})}$. In [31], the authors introduced a $\frac{44}{9}$-factor approximation algorithm for the MDS problem in UDGs, employing a local improvement technique. This algorithm operates with

[1] A bipartite graph $B = (X \cup Y, E)$ is said to be a tree convex bipartite graph if there exists a tree $T = (X, F)$, where X is the set of vertices and F is the set of edges such that for each $v \in Y$, the neighborhood of v induces a sub-tree of T. A tree convex bipartite graph $G = (X, Y, E)$ is a star convex bipartite if T is a star.

[2] A bipartite graph $G = (X, Y, E)$ is circular convex if the vertices in X can be circularly ordered such that for every vertex in $y \in Y$, the neighborhood is an interval of the ordering.

[3] A tree convex bipartite graph $G = (X, Y, E)$ is triad convex if T is a triad.

[4] A tree convex bipartite graph $G = (X, Y, E)$ is triad convex if T is a path.

[5] Circle graphs are the intersection graphs of chords of a circle.

a running time of $O(n \log n)$. Additionally, the authors proposed a $\frac{43}{9}$-factor approximation algorithm for the MDS problem, which is designed to accept the adjacency list of the graph, and has a running time $O(n^2m)$. In 2013, De et al. introduced several approximation algorithms for the MDS problem in UDGs, achieving approximation factors 12, 4 and 3 with time complexities $O(n \log n)$, $O(n^8 \log n)$ and $O(n^{15} \log n)$, respectively.

The literature on various other variants of dominating sets, such as efficient domination, independent dominating sets, k-domination, paired domination, Roman domination, and secure domination, can be found in sources [7, 11, 15, 21, 28, 50, 54, 81, 89].

2.1 Total Dominating Set

Let $G = (V, E)$ be an undirected graph with no isolated vertex. A subset $D_t \subseteq V$ is a total dominating set (TDS) if D_t is a dominating set and for each vertex $u \in D_t$, there exists another vertex $v \in D_t$ such that $uv \in E$. The total dominating set of minimum size is called a minimum total dominating set (MTDS). The cardinality of a minimum total dominating set is called the total domination number ($\gamma_t(G)$). Given graph G, the total dominating set problem (TDSP) aims to find a total dominating set of minimum size.

Existing Results: The concept of *total dominating set* was introduced by Cockayne et al. [27]. They showed that for any connected graph G with at least 3 vertices ($n \geq 3$), the cardinality of *minimum total dominating set* is less than $\frac{2}{3}n$ i.e. $\gamma_t \leq \frac{2}{3}n$. In [9], Atapour and Soltankhah showed that for any graph G with no isolated vertex with maximum degree Δ and n vertices, $\gamma_t \leq n - \Delta + 1$. They also characterized the bipartite graphs and trees that achieve this upper bound. In [36], the authors investigated the relationship between the *upper total domination number* ($\Gamma_t(G)$) and the *upper paired domination number* ($\Gamma_{pr}(G)$) of a graph. They showed that for every graph G with no isolated vertex $\Gamma_t(G) \geq \frac{1}{2}(\Gamma_{pr}(G) + 2)$, and also characterized the trees that achieved this bound. In the same paper, they also

showed that for the family of trees T on at least two vertices, $\Gamma_t(T) \leq \Gamma_{pr}(T)$ holds[1]. Thomassé and Yeo [102] proved that every graph with minimum degree at least 3 (respectively, 4) has *total domination number* at most $\frac{n}{2}$ (respectively, $\frac{3n}{7}$). In [34], DeLaViņa et al. showed that in a connected graph with $n \geq 1$, the *total domination number* is at least the radius of the graph (i.e., $\gamma_t \geq r$)[2]. They also proved that the *total domination number* of any connected graph equals the *total domination number* of a spanning tree of the same graph. Another interesting aspect of trees with respect to total domination is that it is possible to characterize some vertices that are in every *total dominating set* or not in any *total dominating set* [29]. Furthermore, in [55], Haynes and Henning established three equivalent conditions to have a unique *minimum total dominating set* in a tree. The authors also gave a constructive characterization of such trees in the same paper. In [22], Chellali and Haynes characterized the trees for which $\gamma_t = \frac{n+2-l_f}{2}$ and proved that for a non-trivial tree of order n with l_f leaves the *total domination number* is at least $\frac{n+2-l_f}{2}$. In [37], the authors showed that planar graphs with diameter 3 and radius 2 have *total domination number* at most 5.

As far as complexity is concerned, the decision version of *total dominating set problem* is NP-complete, even when restricted to bipartite graphs [94]. However, there is a linear-time algorithm for computing *total dominating set* of a tree [77]. In [77], the authors also showed that the problem remains NP-complete in undirected path graphs. For star graphs, complete graphs, binary star graphs, and complete bipartite graphs, *total domination number* is 2 [6]. In 1993, Keil and Mark proved that the *total dominating set problem* is NP-complete in circle graphs. In the same article, they observed that the *total domination number* can be obtained in polynomial time for cycles and paths. They have also established a set of relations between (i) γ_t and the maximum degree, and (ii) γ_t and the cut vertices of the graph. Peter Damaschke et.al. showed that the MTDS problem is polynomially solvable in chordal bipartite graphs [32]. Recently, in 2021, Jena and Das [72] showed that the TDS

[1]The maximum cardinality of a minimal total dominating set and a minimal paired-dominating set of a graph G is the upper total domination number ($\Gamma_t(G)$) and upper paired-domination number ($\gamma_{pr}(G)$) of the graph G, respectively.

[2]Given a graph G, radius is the minimum eccentricity taken over all the vertices of G

problem in UDGs is NP-complete and gave an 8-factor approximation algorithm that runs in $O(|V| \log |D_t|)$ time, where $|D_t|$ is the size of the output. In the same paper, the authors presented a polynomial time approximation scheme (PTAS) that runs in $O(k^2 n^{2(\lceil 2\sqrt{2}k \rceil)^2})$ time to compute a total dominating set of size at most $(1 + \frac{1}{k})^2 |D_t^*|$, where $k \geq 1$ and D_t^* is the minimum TDS. See [6, 27, 59, 71, 108] for a detailed survey on the TDS problem. Few other variants of total domination can be found in sources [60, 68, 70, 76].

2.2 Semi-total Dominating Set

Let $G = (V, E)$ be an undirected graph with no isolated vertex. A subset $D_{t2} \subseteq V$ is a semi-total dominating set (T2DS) if D_{t2} is a dominating set and for each vertex $u \in D_{t2}$, there exists another vertex $v \in D_{t2}$ such that the distance between u and v is within 2. The semi-total dominating set with minimum size is called a minimum semi-total dominating set (MT2DS), and the correponding cardinality is the semi-total domination number ($\gamma_{t2}(G)$). Given a graph G with no isolated vertices, the semi-total dominating set problem aims to find a semi-total dominating set of G with minimum cardinality.

Existing Results: The concept of *semi-total dominating set* was introduced by W. Goddard et al. in 2014 [44]. Since every *semi-total dominating set* is a dominating set and every *total dominating set* is a *semi-total dominating set*, the *semi-total domination number* is squeezed between the *domination number* and the *total domination number*, i.e., for a given graph G, $\gamma(G) \leq \gamma_{t2}(G) \leq \gamma_t(G)$. In [83], authors showed that if G is a connected graph with $n \geq 4$ vertices, then $\gamma_{t2} \leq \frac{n}{2}$. In the same paper, authors also characterized trees and graphs of minimum degree 2 achieving the bound and showed that if G is a graph with n vertices and maximum degree Δ, then $\gamma_{t2} \geq \frac{2n}{2\Delta+1}$. Subsequently, Henning and Marcon [64] showed that for a connected graph with at least 2 vertices, $\gamma_{t2}(G) \leq \alpha'(G) + 1$, where $\alpha'(G)$ is the *matching number* and then characterized the graphs achieving equality in the bound[1]. They also showed that in a connected graph with at least 4 vertices

[1]Given a graph $G = (V, E)$, a set of edges $E' \subseteq E$ is said to be a matching of G if no two elements of E' are adjacent and the matching number is the size of the largest matching.

(which is not a star), $\gamma_{t2} \leq \alpha'(G)$. In [8], Asplund et al. studied the semi-total domination in cartesian product graphs and established that for any two graphs G and H, $\gamma_{t2}(G\Box H) \geq \frac{1}{3}\gamma_{t2}(G)\gamma_{t2}(H)$. They also proved that for any two graphs G_1 and G_2 without any isolated vertices, $\gamma_{t2}(G_1\Box G_2) \geq \rho(G_1)\gamma_{t2}(G_2)$. In [74], Zeliha Kartal and Aysun Ayta studied the semi-total domination number in Harary graphs [1]. In [61], authors established that if G is a connected graph with minimum degree $\delta \geq 1$ and of order $n \geq \delta + 2$, then $\gamma_{t2}(G) \leq n - \delta$, and the bound is sharp for every fixed $\delta \geq 1$. In [43], authors showed that it is NP-complete to recognize the graphs that satisfy $\gamma_{t2}(G) = \gamma_t(G)$ and $\gamma(G) = \gamma_{t2}(G)$. In the same paper, the authors showed that the problem is solvable in polynomial time for the class of graphs of bounded min-width through a reduction to the *total dominating set*. They also provided some approximation lower bounds for sub-classes of sub-cubic graphs. In [65], authors proved that for every connected claw-free cubic graph G of order n (≥ 10), $\gamma_{t2} \leq \frac{4n}{11}$. In [69], authors showed that the *semi-total domination problem* remains NP-complete in planar, chordal bipartite, and split graphs. They also gave a polynomial time algorithm for the *semi-total domination problem* in interval graphs and a $2 + 3\ln(\Delta + 1)$-factor approximation algorithm for general graphs, where Δ is the degree of G. The authors showed that the minimum semi-total domination problem cannot be approximated within $(1-\epsilon)\ln n$ for any $\epsilon > 0$ unless $NP \subseteq DTIME(n^{O(\log\log n)})$. Finally, In the same paper, the authors also proved that the problem is APX-complete for bipartite graphs with maximum degree 4. It is also further studied in [24, 56, 66, 67, 101].

2.3 Total Roman Dominating Set

Let $G = (V, E)$ be an undirected graph with no isolated vertex. The total Roman dominating set (TRDS) is an ordered partition of V, say (V_0, V_1, V_2) induced by a function, $f : V \to \{0, 1, 2\}$ called total Roman dominating function (TRDF) such that (i) f is a Roman dominating function (Roman property), and (ii) the induced subgraph $G[V_1 \cup V_2]$ does not contain any isolated vertex (total property). The TRDF with minimum weight,

[1] Harary graphs are the k-connected graphs of n vertices with the smallest possible number of edges.

$f(V) = \sum_{u \in V} f(u)$, is called the total Roman domination number and is denoted by $\gamma_{tR}(G)$. Given graph G, the total Roman dominating set problem aims to find a total Roman dominating set of minimum size.

Existing Results: The concept of a *total Roman dominating set* (TRDS) is an extension of the *Roman dominating set* (RDS) with additional constraints that ensure stronger coverage and connectivity properties. The concept of a *Roman Dominating Set* (RDS) was first introduced by Cockayne et al. [28], motivated by an article by Stewart [100], which discussed legion deployment for enhanced security using limited resources. Some of the properties/results of RDS listed in [28] are: (i) for any graph G, $\gamma(G) \leq \gamma_R(G) \leq 2\gamma(G)$, (ii) For any graph G of order n, $\gamma(G) = \gamma_R(G)$ if and only if $G = K_n$, (ii) For any non-trivial connected graph G, $\gamma_R(G) = \min\{2\gamma(G \setminus \mathcal{P}) + |\mathcal{P}| : \mathcal{P}$ is a 2-packing$\}$[1]. RDS problem is NP-complete for general graphs [38]. It is also NP-complete when restricted to chordal, bipartite graphs, split graphs, and planar graphs, as mentioned in [28]. Additional literature and variants on the RDS problem can be found in works such as [1, 2, 40, 80, 96, 97] etc.

One variant of the *Roman dominating set* (RDS) is the *total Roman dominating set* (TRDS), which was first introduced by Liu and Chang [80]. They demonstrated that the TRDS problem is NP-complete in various types of graphs, including bipartite graphs, chordal graphs, and split graphs. In [3], authors established lower and upper bounds on *total Roman dominating set* and related the *total Roman domination number* ($\gamma_{tR}(G)$) to domination parameters such as the *domination number* ($\gamma(G)$), *Roman domination number* ($\gamma_R(G)$) and *total domination number* ($\gamma_t(G)$). They showed that for an isolate-free graph G, $\gamma_{tR}(G) \leq 3\gamma(G)$. Further, if $\gamma_{tR}(G) = 3\gamma(G)$, then every *minimum dominating set* is a packing in G[2]. In [85], authors introduced a new lower and upper bound for $\gamma_{tR}(G)$, which was even tighter than the well known bound, $2\gamma(G) \leq \gamma_{tR}(G) \leq 3\gamma(G)$. They proved that for any graph G with neither isolated vertex nor components isomorphic to K_2, $\gamma_{t2}(G) + \gamma(G) \leq \gamma_{tR}(G) \leq$

[1]A subset $S \subseteq V(G)$ is a 2-packing of G, if for any two distinct vertices u and v, $d(u, v) > 2$, and the largest cardinality of a 2-*packing* of the graph G is the packing number of G and is denoted by $\rho(G)$.

[2]A packing in G is a set of vertices that are pairwise at distance at least 3.

$\gamma_R(G) + \gamma(G)$ holds. Furthermore, they established that in an isolate-free graph G, the equality $\gamma_{tR}(G) = 2\gamma(G)$ holds if and only if $\gamma_{tR}(G) = \gamma_{t2}(G) + \gamma(G)$ and $\gamma_{t2}(G) = \gamma(G)$. In 2020, A. Poureidi [95] gave a linear-time algorithm to compute the *total Roman domination number* for proper interval graphs in $O(|V|)$ time. For general graphs, more results and variations on TRDS can be found in [5, 47, 99, 104]. Here are some related works specifically focusing on trees. In [5], the authors observed that if T is a star of order at least 2, then $\gamma_{tR}(T) < 2\gamma_t(T)$. Furthermore, they provided a constructive characterization of all trees satisfying $\gamma_{tR}(T) = 2\gamma_t(T)$. In a more recent work in 2021, authors in [84] offered a constructive characterization of trees satisfying $\gamma_{tR}(T) = \gamma_t(T) + \gamma(T)$. It is also further studied in [18, 86, 87, 90]

Chapter 3

Semi-total Domination in UDGs and General Graphs

The concept of *semi-total dominating set* is a significant topic in combinatorial optimization problems in graph theory, primarily because of its wide range of applications. In this chapter, we study the *semi-total dominating set* (T2DS) problem in unit disk graphs (UDGs) and general graphs.

Definition 1. *Given an undirected simple graph $G = (V, E)$ with no isolated vertex. A subset $D_{t2} \subseteq V$ is a semi-total dominating set (T2DS) if D_{t2} is a dominating set and for each vertex $u \in D_{t2}$, there exists another vertex $v \in D_{t2}$ such that $d(u, v) \leq 2$, where $d(u, v)$ is the minimum number of edges connecting u with v. The semi-total dominating set with minimum size is called a minimum semi-total dominating set, and the correponding cardinality is the semi-total domination number ($\gamma_{t2}(G)$). Given a graph G, a semi-total dominating set problem aims to find a semi-total dominating set of minimum size.*

Definition 2. (Semi-total dominating set problem in general graphs) *Given an undirected simple graph $G = (V, E)$ with no isolated vertex, find a semi-total dominating set $D_{t2} \subseteq V$ of minimum size.*

Definition 3. (Semi-total dominating set problem in UDGs) *Given a unit disk graph $G = (V, E)$ corresponding to a point set $P = \{p_1, p_2, \ldots, p_n\}$ representing the disk centers*

in the plane, find a semi-total dominating set $D_{t2} \subseteq V$ *of minimum size.*

The goals of the chapter are:

- to prove that the *semi-total dominating set problem* in UDGs is NP-complete
- to obtain a 6-factor approximation algorithm for the *semi-total dominating set problem* in UDGs
- to obtain a $2 + \ln(\Delta + 1)$-factor approximation algorithm for the *semi-total dominating set problem* in general graphs.

The remainder of the chapter is organized as follows. In Section 3.1, we listed the required preliminaries. In Section 3.2, we prove the hardness result of the *semi-total dominating set problem* in UDGs. Furthermore, in Section 3.3 and Section 3.4, we propose a 6-factor and a $2 + \ln(\Delta + 1)$-factor approximation algorithms for the UDGs and general graphs, respectively, and finally conclude the chapter in Section 3.5.

3.1 Preliminaries

In this section, we introduce the required notations and definitions. Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of n points in $\mathbb{R}^2$. A graph $G = (V, E)$ is said to be a geometric UDG corresponding to the point set P if there exists a one-to-one correspondence between each $v_i \in V$ with $p_i \in P$ and $v_i v_j \in E$ if and only if $\delta(p_i, p_j) \leq 1$, where $\delta(.,.)$ is the Euclidean distance between two points in $\mathbb{R}^2$. Let $U(p)$ denote the unit disk centered at the point $p \in P$ and $U(P) = \{U(p) : p \in P\}$. The set of disks $U(P)$ is considered independent if for every pair $p, q \in P$, $p \notin U(q)$, i.e., $\delta(p, q) > 1$. Here, $d(p, q)$ refers to the number of edges on the shortest path between p and q. In this chapter, we often refer to a node or vertex as a 'point' since we are working within the context of geometry.

Next, we present a compilation of previously established lemmas, theorems, and observa-

tions that will be useful in Section 3.2, Section 3.3 and Section 3.4.

In [103], L. Valiant addressed the problem of embedding planar graphs into a grid. The goal of this paper was to map a planar graph onto a grid such that no edges cross and the area of the grid is minimized. The author presented an algorithm that constructs a planar embedding of a graph on an integer grid with minimal area. The author demonstrated that there exist two constants $c_1, c_2 > 0$ such that for a planar graph with n vertices, the area $\mathcal{A}$ of the embedding satisfies $c_1 n^2 \leq \mathcal{A} \leq c_2 n^2$. This implies that the area required for embedding scales quadratically with the number of vertices. We recreated the corresponding theorem from [103] and present it here as a lemma relevant to our context.

Lemma 3.1.1. *[103] Let $G = (V, E)$ be a planar graph of degree at most 3. The graph G can be embedded in a grid of area $O(|V|^2)$ such that each $v \in V$ is positioned at a grid point with co-ordinates $(5i, 5j)$, where i and j are integers, and each edge $e \in E$ is a finite sequence of consecutive segments, each of length 5 units, aligned along the grid lines.*

Lemma 3.1.2. *[82] Let $\mathcal{P}$ be a unit disk centered at point p and let S be a set of independent unit disks such that each disk in S contains the point p. Then, $|S| \leq 5$.*

Proof. Assume, on the contrary, that $|S| >= 6$. Let c_i represent the centers of any 6 circles in S, where $1 \leq i \leq 6$. Since each disk in S contains the point p, we have $\delta(p, c_i) \leq 1$ for $1 \leq i \leq 6$. Let $\overrightarrow{pc_i}$ be the ray from p to c_i. With 6 rays originating from p, by pigeon hole principle, there must exist at least one pair of rays (say, $\overrightarrow{pc_i}$ and $\overrightarrow{pc_j}$) such that the angle formed by them at p is at most $\frac{\pi}{3}$, which implies $\delta(c_i, c_j) \leq 1$, meaning the corresponding disks overlap and are not independent. Thus, $|S| \leq 5$. □

In [31], authors achieved a $\frac{44}{9}$-factor approximation for the *Minimum Dominating Set* (MDS) problem in UDGs. The corresponding algorithm partitions the graph into grid cells of side length $\frac{1}{3}$ to localize the problem. Within each cell, a *maximal independent set* is computed to ensure domination in that region. These local solutions are combined and refined to form a global dominating set while maintaining the domination property across the entire graph. The algorithm guarantees that the size of the resulting *dominating set* is at most $\frac{44}{9}$ times

the size of the optimal solution. This algorithm runs in $O(n+m)$ time, where n and m are the numbers of vertices and edges of the given UDG. We summarize the above algorithm in the following lemma.

Lemma 3.1.3. *[31] Given a UDG G, there exists a $\frac{44}{9}$-approximation algorithm for the minimum dominating set problem with running time $O(n^2)$, where n is the number of vertices in the given UDG.*

Observation 3.1.1. *[44] For a given graph G, $\gamma(G) \leq \gamma_{t2}(G)$, where $\gamma(G)$ and $\gamma_{t2}(G)$ are the domination number and the semi-total domination number of the graph G, respectively.*

3.2 NP-completeness Result

In this section, we focus on the hardness result of the *semi-total domination problem* and prove that the decision version of the problem is NP-complete in unit disk graphs. We achieve this by reducing the decision version of the *vertex cover problem* (VCP) in planer graphs of degree at most 3 to the decision version of the *semi-total dominating set problem* in UDGs. The corresponding decision problems are formally defined as follows:

The decision version of the VC problem in planar graphs of degree at most 3 (D-VC-PGD3): Given a positive integer k and a planar graph G of degree at most 3, does G have a VC of size at most k?

The decision version of the semi-total dominating set problem in UDGs (D-T2DS-UDGs): Given a positive integer k and a UDG G, does G have a *semi-total dominating set* of size at most k?

Lichtenstein and David [78] reduced the *planar 3SAT problem* to the *planar vertex cover* problem and proved that **D-VC-PGD3** is NP-complete. We prove the hardness result of the *semi-total dominating set problem* in UDGs by showing a polynomial time reduction from an arbitrary instance of **D-VC-PGD3** to an instance of **D-T2DS-UDGs**. To prove

this, we embed a planar graph $G = (V, E)$ of degree at most 3 in a grid of cell size 5×5 using Lemma 3.1.1.

Lemma 3.2.1. *If $G = (V, E)$ is an instance of **D-VC-PGD3** without any isolated vertex, then an instance $G' = (V', E')$ of **D-T2DS-UDG** can be constructed from G in polynomial time.*

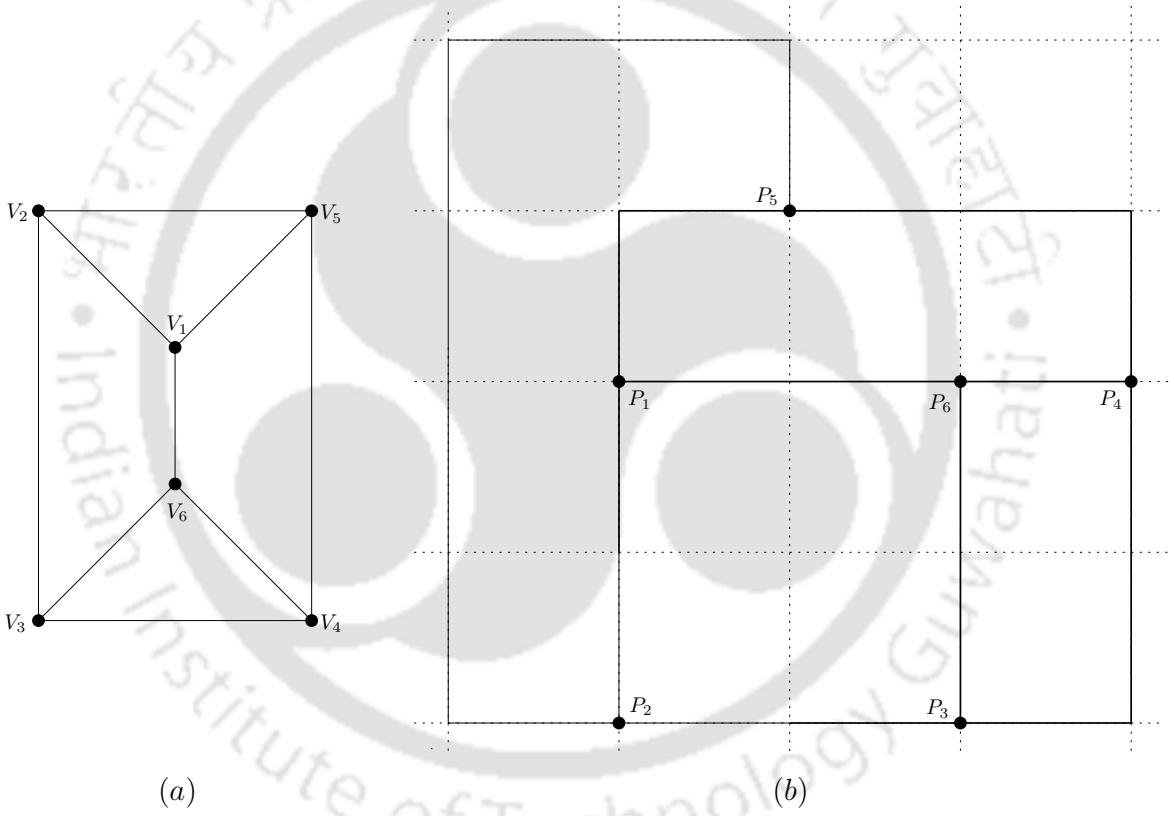

Figure 3.1: *(a) A planar graph G, and (b) Embedding of G in a grid*

Proof. Let $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$ be the vertex set and edge set of the given instance $G = (V, E)$, where $|V| = n$ and $|E| = m$. We construct a graph $G' = (V', E')$ from G by performing the four steps as mentioned below:

Step 1 **(Embedding):** First, we embed the graph G on a grid of cell size 5×5 using

the algorithm proposed by Biedl et al. in [14][1]. In this embedding, each edge $e \in E$ is represented as a consecutive sequence of line segment(s) in the grid, where each segment has a length of 5 units. Let ℓ denote the total number of line segments used in the embedding. Each vertex $v \in V$ is mapped to a node point located at the grid coordinates $(5i, 5j)$, where i and j are integers. For each vertex $v_i \in V$, the corresponding node point is denoted as p_i, where $1 \leq i \leq n$. We refer to these node points as **embedding nodes**. Let N be the set of these nodes, where $|N| = n$. Refer to Figure. 3.1(a) and Figure. 3.1(b) for an illustration of the embedding step.

Step 2 (Inclusion of auxiliary nodes): In this step, we add some auxiliary nodes on each segment of the graph after the embedding step as given below:
(*i*) *Multiple Segments Edges:* For edges represented by multiple segments (length greater than 5 units), first, we add a node at each grid point along the segments, excluding the node points (embedding nodes). We refer to these points as **grid nodes** (see the filled square points in Figure. 3.2). Next, we place the nodes along the segments as per the criteria mentioned below. These are reffered as **auxiliary nodes**.

- If both endpoints of a segment are grid nodes, add four nodes on the segment at distances of 1, 2, 3, and 4 units from either endpoint, as shown in Figure. 3.3(c).
- If one endpoint is an embedding node p_i and the other one is a grid node, add five nodes on the segment at distances of 1, 1.9, 2.5, 3 and 4 units from p_i, as depicted in Figure. 3.3(b).

(*ii*) *Single Segment Edges:* For each edge $v_iv_j \in E$ represented by a single segment p_ip_j (with a length of exactly 5 units), add *six* nodes at distances 1, 1.3, 2.1, 2.6, 3.2 and 4 units from either p_i or p_j as depicted in Figure. 3.3(a).
Let A be the set that represents the auxiliary nodes added in this step, where $|A| = 4\ell + m$.

[1]A planar graph embedding in the Euclidean plane can be constructed by starting with a single vertex and incrementally adding vertices and their connecting edges along the outer boundary of the structure developed so far, which avoids edge crossings if done carefully, respecting the planar structure. This approach ensures that each vertex is placed at a grid point and each edge is represented as a sequence of consecutive segments along the grid lines. The total number of such segments used in the embedding is ℓ.

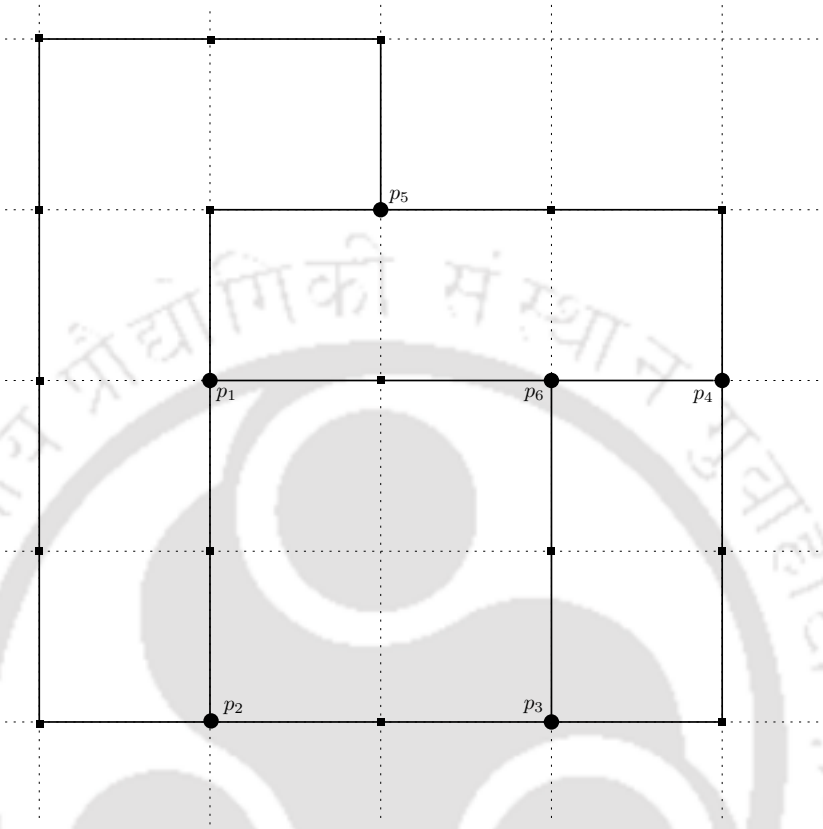

Figure 3.2: *Inclusion of grid nodes*

Step 3 **(Inclusion of gadgets):** Since each node in the planar graph has degree at most 3 and is embedded within a grid, there is at least one position at each embedding node to accommodate an extra edge. In this step, we introduce a gadget at each embedding node p_i as shown in Figure. 3.3(d). The gadget at p_i contains 4 nodes, namely x_i, x'_i, y_i and y'_i. Here, the distances between p_i and x_i, x_i and x'_i, x_i and y_i, y_i and y'_i are 0.9, 0.5, 0.9 and 0.5, respectively. Let S be the set of nodes added in this step. Since each gadget introduces 4 new nodes and we add one gadget for each embedding node, the total number of nodes added in this step is $|S| = 4|N| = 4n$.

Step 4 **(Construction of UDG):** Let $G' = (V', E')$ be the UDG constructed after applying Steps 1 - 3 on graph G, where $V' = N \cup A \cup S$ and $E' = \{uv : u, v \in V' \text{ and } \delta(u, v) \leq 1\}$.

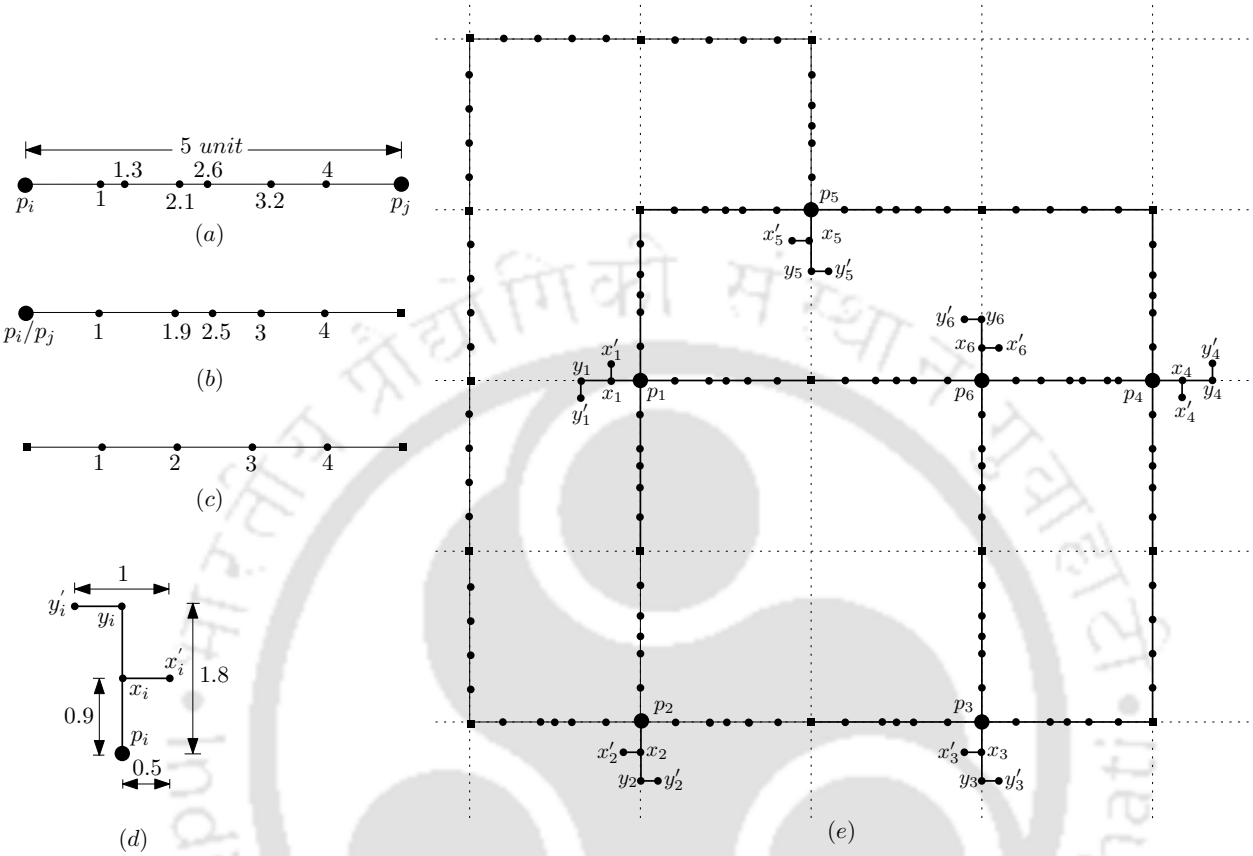

Figure 3.3: *(a) Orientation of six nodes, (b) Orientation of five nodes, (c) Orientation of four nodes, (d) Gadget, and (e) Graph G'*

From Lemma 3.1.1, we conclude that the number of segments $\ell = O(n^2)$. Therefore, the number of vertices and the number of edges in G' is $O(n^2)$. Hence, G' can be constructed from G in polynomial time. For the complete illustration of the construction phases, refer to Figure. 3.1 and Figure. 3.3. □

Theorem 3.2.1. ***D-T2DS-UDGs*** *belongs to the class NP-complete.*

Proof. Let $G = (V, E)$ be a unit disk graph. Given a subset $D_{t2} \subseteq V$ and a positive integer k, we can verify whether D_{t2} is a *semi-total dominating set* of G of size at most k or not in polynomial time. Therefore, $D\text{-}T2DS\text{-}UDGs \in NP$.

To prove $D\text{-}T2DS\text{-}UDGs$ is NP-hard, we perform a polynomial time reduction from

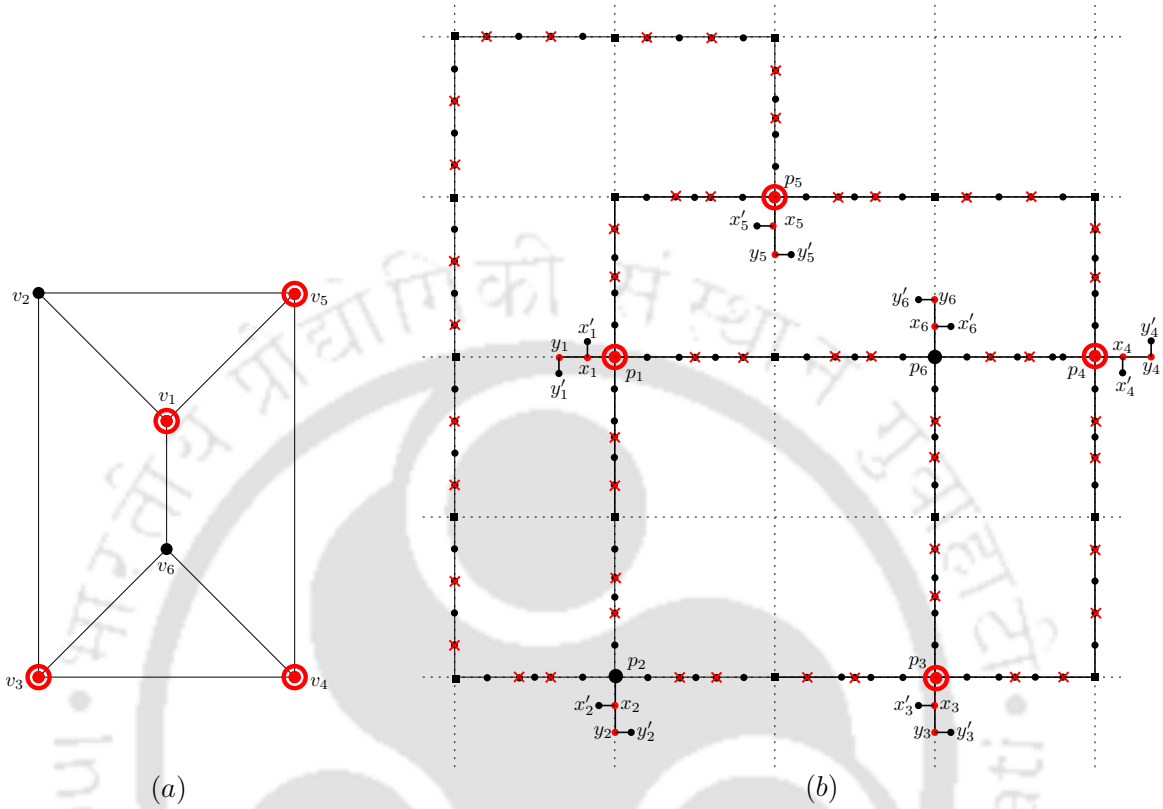

Figure 3.4: *(a) Vertex cover of G, and (b) Semi-total dominating set of G'*

D-VC-PGD3 to *D-T2DS-UDGs*. Using Lemma 3.2.1, We construct an instance $G' = (V', E')$ of *D-T2DS-UDGs* from an arbitrary instance $G = (V, E)$ of *D-VC-PGD3* in polynomial time. From Claim 3.2.1.1, *D-T2DS-UDGs* $\in$ *NP-hard*.
Therefore, *D-T2DS-UDGs* $\in$ *NP-complete*. □

Claim 3.2.1.1. *G has a vertex cover S_{vc} such that $|S_{vc}| \leq k$ if and only if G' has a* semi-total dominating set *D_{t2} such that $|D_{t2}| \leq k + 2\ell + 2n$.*

Proof. **Forward Direction** ($\Longrightarrow$)**:** Let S_{vc} be a *vertex cover* of G such that $|S_{vc}| \leq k$. Let T_{vc} be the set of vertices in G' corresponding to the vertices in S_{vc}, i.e., $T_{vc} = \{p_i \in V' : v_i \in S_{vc}\}$. Now, we construct two sets $T_a \subseteq A$ and $T_g \subseteq S$ such that $D_{t2} = T_{vc} \cup T_a \cup T_g$ is a *semi-total dominating set* with cardinality less than or equal to $k + 2\ell + 2n$. The constructions of T_a and T_g are as follows. Since S_{vc} is a *vertex cover* of G, at least one endpoint of every edge in G is inside S_{vc}. Since every edge in G corresponds to a sequence of segments

in G', we start from the endpoint, which is inside T_{vc}. For each p_ip_j in G' corresponding to each $v_iv_j \in E$, at least one out of p_i and p_j is in T_{vc}. Without loss of generality, let $p_i \in T_{vc}$. We traverse from p_i to p_j. During traversal, we skip two vertices next to p_i, add a single vertex to T_a, then skip the next vertex and select the following one for T_a; we repeat this process until we reach p_j. For each $v_iv_j \in E$, we apply this process to the corresponding p_ip_j in G' and observe that exactly two points from each segment are in T_a. So $|T_a| = 2\ell$, where ℓ is the number of segments in G'. In Figure. 3.4(b), the red cross points are in T_a. From each $p_i \in V'$, we choose x_i and y_i for T_g. So $|T_g| = 2n$. In Figure. 3.4(b), the red disks are in T_g.

Now, we are left to show that the set D_{t2} is a *semi-total dominating set* of G'. To demonstrate this, we need to ensure that each vertex in G' is either in D_{t2} or adjacent to a vertex in D_{t2}, and for each vertex $u \in D_{t2}$ there exists another vertex v such that $d(u, v) \leq 2$. To show this, recall the definitions and properties of the sets involved:

- $x_i, y_i \in T_g \subseteq D_{t2}$
- x_i and y_i dominate themselves, and p_i, x'_i and y'_i
- $d(x_i, y_i) \leq 2$ for all $i = \{1, 2, \ldots, n\}$
- for each $p_i \in T_{vc}$, there exists $x_i \in T_g$ such that $d(p_i, x_i) \leq 2$

These properties ensure that the sets T_{vc} and T_g meet the semi-total domination requirements for the vertices p_i, x_i, y_i, x'_i and y'_i in G'. Now, we need to show that the remaining vertices in G' also satisfy the semi-total domination requirements, and this is done by analyzing each type of segment in G'. There are three types of segment in G'. The types of segments are as follows: (i) segments with two endpoints as node points (refer to Figure. 3.3(a)), (ii) segments with one endpoint as node point and another as grid point (refer to Figure. 3.3(b)), and (iii) segments with two endpoints as grid points (refer to Figure. 3.3(c)). We have to show that each segment from each segment type requires at least two vertices in D_{t2} for semi-total domination.

(i) **Segments with two endpoints as node points:** Let us consider a segment $v_iv_j \in E$, where at least one of v_i and v_j is in S_{vc} because S_{vc} is a *vertex cover* of G. Without loss of generality, assume $v_i \in S_{vc}$. This implies $p_i \in T_{vc}$. The structure of G' for this type of segment is such that there are six intermediate points between the node points p_i and p_j, which we label as z_t^{ij} for $1 \leq t \leq 6$, where t indicates the position of the point from p_i. The specific positions and their domination relationships are as follows:

- p_i dominates z_1^{ij}
- z_3^{ij} and z_5^{ij} are selected in T_a to ensure that z_3^{ij} dominates z_2^{ij} and z_4^{ij}, and z_5^{ij} dominates z_6^{ij}

Since $d(z_3^{ij}, z_5^{ij}) \leq 2$ and $d(p_i, x_i) \leq 2$, the *semi-total property* is satisfied in p_ip_j. Thus, the set of vertices $\{p_i, z_3^{ij}, z_5^{ij}\}$ preserves the semi-total domination properties for the segment p_ip_j (refer to the segment p_4p_6 in Figure. 3.4(b)).

(ii) **Segments with one endpoint as a node point and other one as a grid point:** To demonstrate this, we analyze the two possible cases: (a) $p_i \in T_{vc}$ and (b) $p_i \notin T_{vc}$. Let us break down these cases:

Case (a): $p_i \in T_{vc}$

Given a segment p_ig^{ij} where p_i is the node point and g^{ij} is the grid point. The structure of G' for this type of segment is such that there are five intermediate points. Let the points are z_1^{ij}, z_2^{ij}, z_3^{ij}, z_4^{ij} and z_5^{ij}, where z_t^{ij} represents the point present at t^{th} position from p_i. The specific position of the points and their domination relationships are as follows:

- p_i dominates z_1^{ij}.
- selecting z_3^{ij} and z_5^{ij} in T_a ensures that z_3^{ij} dominates z_2^{ij} and z_4^{ij}, z_5^{ij} dominates g^{ij}.
- the distance $d(z_3^{ij}, z_5^{ij}) \leq 2$ ensures the semi-total property along the segment.

Thus, the set of vertices $\{p_i, z_3^{ij}, z_5^{ij}\}$ is a semi-total dominating set for the segment p_ig^{ij}.

Case (b): $p_i \notin T_{vc}$

If $p_i \notin T_{vc}$, then $p_j \in T_{vc}$ because S_{vc} is a *vertex cover*. The segment $p_i g^{ij}$ contains the same five points z_1^{ij}, z_2^{ij}, z_3^{ij}, z_4^{ij} and z_5^{ij}, and the domination relationship among the points are as follows.

- Since $p_j \in T_{vc}$, there exists a point on the segment with g^{ij} as one of the endpoints that dominates g^{ij} (evident from Case (a)).
- Selecting z_2^{ij} and z_4^{ij} in T_a ensures that z_2^{ij} dominates z_1^{ij} and z_3^{ij}, z_4^{ij} dominates z_5^{ij}.
- The distance $d(z_2^{ij}, z_4^{ij}) \leq 2$ ensures the semi-total property along the segment.

This configuration similarly ensures that the set of vertices $\{z_2^{ij}, z_4^{ij}\}$ is a *semi-total dominating set* for the segment $p_i g^{ij}$.

(*iii*) **Segments with two endpoints as grid points:** Let us consider a segment $g_b^{ij} g_{b+1}^{ij}$, where g_b^{ij} and g_{b+1}^{ij} are two grid points (endpoints). The structure of G' for this type of segment is such that there are four intermediate points between the two grid points, which we label as $z_t^{b(b+1)}$ for $1 \leq t \leq 4$, where t indicates the position of the point from g_b^{ij}. Given that at least one of p_i or p_j is in T_{vc} because S_{vc} is a *vertex cover*, at least one grid point is dominated by a point in the other segment connected either to g_b^{ij} or g_{b+1}^{ij} as evident from Case (a) of (ii). Without loss of generality, assume g_{b+1}^{ij} is the grid point which is dominated by a point present on the segment connected to the segment $g_b^{ij} g_{b+1}^{ij}$. Then the domination relationship among the points on the segment is given below.

- Selecting $z_2^{b(b+1)}$ and $z_4^{b(b+1)}$ in T_a ensures that $z_2^{b(b+1)}$ dominates $z_1^{b(b+1)}$ and $z_3^{b(b+1)}$, $z_4^{b(b+1)}$ dominates $z_5^{b(b+1)}$.
- The distance $d(z_2^{b(b+1)}, z_4^{b(b+1)}) \leq 2$ ensures the *semi-total property* along the segment.

Hence, the set of vertices $\{z_2^{b(b+1)}, z_4^{b(b+1)}\}$ is a *semi-total dominating set* for the points present on the segment $g_b^{ij} g_{b+1}^{ij}$.

From the above arguments, we conclude that $D_{t2} = T_{vc} \cup T_a \cup T_g$ is a semi-total dominating set of G'. Since $|T_{vc}| \leq k$, $|T_a| = 2l$ and $|T_g| = 2n$, $|D_{t2}| \leq k + 2\ell + 2n$.

Reverse Direction ($\Longleftarrow$)**:** Let D_{t2} be a *semi-total dominating set* of G' such that $|G_{t2}| \leq k + 2\ell + 2n$. Then, we will show that G has a *vertex cover* of size at most k. To achieve this, we will prove the following observations and use them to construct a *vertex cover* of G.

(i) Out of four points in the gadget associated with each p_i in G', at least two points belong to D_{t2}.

(ii) Each segment contributes at least two points to D_{t2}.

(iii) If p_i and p_j in G' correspond to the endpoints of an edge $v_iv_j \in E$ and neither p_i nor p_j is in D_{t2}, then there exists a segment from which 3 vertices are in D_{t2}.

Observation (i): Each gadget associated with p_i in G' consists of four points x_i, x'_i, y_i, and y'_i. For semi-total domination, at least two of these points must be in D_{t2} to ensure domination of x'_i and y'_i (pendant vertices) and the *semi-total property*. Since x'_i and y'_i are pendant vertices, the selection of any two vertices is sufficient for domination. Hence, $|S \cap D_{t2}| \geq 2n$, where S is the set of points in all gadgets.

Observation (ii): For a segment in G' with four intermediate points (excluding the endpoints), at least two points are needed in D_{t2} to ensure semi-total domination. For segments with more points, this observation holds, as domination requires at least two points per segment. Thus $|D_{t2} \cap A| \geq 2\ell$, where A is the set of auxiliary points, and ℓ is the number of segments present in G'.

Observation (iii): Let ℓ' be the number of segments in G' corresponding to an edge $v_iv_j \in E$. Since only consecutive points are adjacent, two points can semi-totally dominate at most 5 points. Therefore, the minimum number of points required to semi-totally dominate $5\ell' + 1$ points on ℓ' segments is $\left\lceil \frac{5\ell'+1}{5} \right\rceil \times 2 = 2\ell' + 1$. Therefore, there exists exactly one segment among these ℓ' segments where three of its points are included in D_{t2}.

Given a *semi-total dominating set* D_{t2} of G' of size at most $k + 2\ell + 2n$, we are left to

show that by deleting and/or replacing some of the vertices from D_{t2}, we can obtain a vertex cover S_{vc} of G of size at most k. Let us define a set $S'_{vc} = D_{t2} \setminus S$, then $|S'_{vc}| \leq k+2\ell$ (due to Observation (i)). Then $S_{vc} = \{v_i \in V | p_i \in S'_{vc}\}$. For each edge $v_iv_j \in E$, if $v_i, v_j \notin S_{vc}$, then there exists a segment on p_ip_j in G', which has 3 points in D_{t2} instead of 2 (refer to Observation (ii) and (iii)). For each such edge, add v_i (or v_j) to S_{vc}. Since every segment contributes at least 2 to D_{t2} and there is 2ℓ number of such points (refer to Observation (ii)), $|S_{vc}| \leq k$. Since every edge in G has at least one vertex in S_{vc}, S_{vc} is a *vertex cover* of size at most k. This proves $D\text{-}T2DS\text{-}UDGs \in NP\text{-}hard$. □

3.3 A 6-factor Approximation Algorithm

In this section, we propose a 6-factor approximation algorithm for the *semi-total domination problem* in UDGs.

3.3.1 Algorithm

Given a geometric unit disk graph $G = (V, E)$ with $V = \{p_1, p_2, \ldots, p_n\} \subseteq \mathbb{R}^2$ as the set of disk centers, Algorithm 3.1 finds a *semi-total dominating set* D_{t2} of G. Now, we describe the procedure for finding the set D_{t2}. First, we find a *maximal independent set* $D \subseteq V$ of G to satisfy the *domination property* (see Lines 2-6 of Algorithm 3.1). Next, to satisfy the *semi-total property*, we choose a set of vertices $T \subseteq V$ such that for each $v \in D$, there exists a vertex $u \in D \cup T$ such that $d(u, v) \leq 2$. To find such a set T, identify each point $u \in D$ that satisfies the *semi-total property* (see Lines 8-13 of Algorithm 3.1) and next, segregate the points (set $\mathcal{U}$) that do not satisfy the *semi-total property* in D (see Line 14 of Algorithm 3.1) and then for each point $u \in \mathcal{U}$, add a point $v \in N_G(u)$ into T (see Lines 15-18 of Algorithm 3.1). Finally, report $D_{t2} = D \cup T$ as a *semi-total dominating set* of G. Lemma 3.3.1 and Lemma 3.3.2 represent the algorithm's correctness and time complexity, respectively.

Lemma 3.3.1. *The set D_{t2} in Algorithm 3.1 is a* semi-total dominating set *of G.*

Algorithm 3.1 T2DS-UDG

Input: A unit disk graph, $G = (V, E)$, with known disk centers
Output: A semi-total dominating set D_{t2} for G

1: $V' = V, D = \emptyset$
2: **while** $V' \neq \emptyset$ **do**
3: choose a vertex $v \in V'$
4: $D = D \cup \{v\}$
5: $V' = V' \setminus N_G[v]$
6: **end while**
7: $T = \emptyset$, $X = \emptyset$
8: **for** each $u \in V$ **do**
9: $S_u = N_G(u) \cap D$
10: **if** $|S_u| > 1$ **then** ▷ each vertex in S_u satisfies the *semi-total* property
11: $X = X \cup S_u$
12: **end if**
13: **end for**
14: $\mathcal{U} = D \setminus X$ ▷ vertices in $\mathcal{U}$ do not satisfy the *semi-total* property
15: **for** each $u \in \mathcal{U}$ **do**
16: choose a vertex $v \in N_G(u)$
17: $T = T \cup \{v\}$
18: **end for**
19: $D_{t2} = D \cup T$
20: **return** D_{t2}

Proof. In the first phase, we find a *maximal independent set* D of G to satisfy the domination property (see Lines 2-6 in Algorithm 3.1). Next, we segregate the points that do not satisfy *semi-total property* in D. Note that for each vertex $u \in V$, the algorithm finds $S_u = N_G(u) \cap D$. If $|S_u| > 1$, then the vertices in S_u satisfy *semi-total property*, and hence the vertices in the set $X \subseteq D$ also satisfy *semi-total property* (see Lines 8-13). Since $\mathcal{U} = D \setminus X$, the vertices in the set $\mathcal{U}$ do not satisfy *semi-total property*. To address this, we choose a one-distance neighbor $v \in V \setminus D$ for each such $u \in \mathcal{U}$ and the set T consists of these corresponding neighbors (see Lines 15-18). Since T includes these one-distance neighbors of each vertex in D that violate the *semi-total property*, the inclusion of T in D_{t2} along with D ensures that for each vertex $u \in D$, there exists another vertex $v \in D \cup T$ such that $d(u, v) \leq 2$. Therefore, the combined sets D and T satisfy both the *domination* and *semi-total* properties. Hence, D_{t2} is a *semi-total dominating set* of G. □

Lemma 3.3.2. *Algorithm 3.1 runs in $O(n \log k)$ time.*

Proof. The complexity of Algorithm 3.1 is primarily dominated by the *three for* loops (see Lines 2-6, Lines 8-13 and Lines 15-18 of Algorithm 3.1). Let $V = \{p_1, p_2, \ldots, p_n\}$ be the set of disks' centers corresponding to graph $G = (V, E)$. Let all the disks lie on a plane's rectangular region $\mathbb{R}$. Let the rectangle's extreme left and bottom arms represent the x- and y-axis, respectively. Then, we split the plane $\mathbb{R}$ so that the region $\mathbb{R}$ becomes a grid with cell size 1×1. Let $[x, y]$ be the index associated with each cell, where $x, y \in \mathbb{N} \cup \{0\}$. If a point $p \in V$ is located at co-ordinate (p_x, p_y) on $\mathbb{R}$, then the point belongs to a cell with index $[\lfloor p_x \rfloor, \lfloor p_y \rfloor]$.

In the first *for* loop (see Lines 2-6), Algorithm 3.1 constructs a maximal independent dominating set D of the input graph G. To do so efficiently, each non-empty cell maintains a list that keeps the points of V chosen for inclusion in D located within that cell. While considering a point $p \in V$ as a candidate for the set D, it only probes into the 9 cells surrounding the cell where p lies. That means if p is located at co-ordinate (p_x, p_y), then it searches in each $[i, j]$ cell, where $\lfloor p_x \rfloor - 1 \leqslant i \leqslant \lfloor p_x \rfloor + 1$ and $\lfloor p_y \rfloor - 1 \leqslant j \leqslant \lfloor p_y \rfloor + 1$.[1] If there does not exist any point $q \in D$ in those 9 cells such that $p \in U(q)$, then p is included in D. A height balance binary tree containing non-empty cells is used to store the points that are in D. Since each cell of size 1×1 can contain the centers of at most 3 independent unit disks (since placing the centers on the boundary maximizes the number of independent unit disks in a cell and one disk covers more than one edge in a cell), the processing time to decide whether a point is in D or not requires $O(\log k)$ time, where $k = |D|$. Thus the time taken to process $|V| = n$ points is $O(n \log k)$.

In the second *for* loop (Lines 8-13), Algorithm 3.1 finds a set $X \subseteq D$ in which each vertex satisfy the *semi-total property*. For each point $p \in V$, it only probes in 9 cells surrounding the cell where p lies, as discussed earlier. It requires $O(\log k)$ time. If there exists any point $q \in D$ in those 9 cells such that $p \in U(q)$, then p is included in S_u. Thus, finding the set X

[1] Any point outside these 9 cells is independent from p.

requires $O(n \log k)$ time.

Since each vertex u in $\mathcal{U}$ does not satisfy *semi-total property* (i.e., $|S_u| \leq 1$), we add a vertex $v \in N_G(u)$ to T (see Lines 15-18). Thus, in the worst case, the time to construct the set T is $O(k)$.

Therefore, in worst case, Algorithm 3.1 executes in $O(n \log k)$ time. □

Lemma 3.3.3. *In Algorithm 3.1, $|T| \leq |D^*|$, where D^* is an optimal DS of G.*

Proof. On contrary assume that $|T| > |D^*|$, which implies $|\mathcal{U}| > |D^*|$. This means that at least one vertex in D^* would have to dominate two or more vertices in $\mathcal{U}$. However, this leads to a contradiction, as it implies there exists a vertex $v \in V$ that has more than one neighbor in $\mathcal{U}$. By the construction of $\mathcal{U}$, no vertex can have more than one neighbor in $\mathcal{U}$. Therefore, $|T| \leq |D^*|$. □

3.3.2 Analysis

The set D_{t2} in Algorithm 3.1 is a *semi-total dominating set* of G, where $D_{t2} = D \cup T$ (see Lemma 3.3.1). Let D^* and D^*_{t2} be the optimal *dominating set* and optimal *semi-total dominating set* of G, respectively. Since D is a *maximal independent set* of G, from Lemma 3.1.2, we have $|D| \leq 5|D^*|$. The set T in Algorithm 3.1 ensures *semi-total property* when added to the *independent set* D. Note that from Lemma 3.3.3, we have $|T| \leq |D^*|$. Therefore, using Lemma 3.1.2, Lemma 3.3.1, Lemma 3.3.3 and Observation 3.1.1, we conclude the approximation factor of Algorithm 3.1 as follows:

$$\begin{aligned} |D_{t2}| = |D \cup T| &\leq |D| + |T| \\ &\leq 5|D^*| + |D^*| \leq 6 \times |D^*| \\ &\leq 6 \times |D^*_{t2}| \end{aligned} \tag{3.1}$$

Theorem 3.3.1. *The proposed algorithm (T2DS-UDG) gives a 6-factor approximation result for the* semi-total dominating set problem *in UDGs. The algorithm runs in $O(n \log k)$*

time, where n is the number of vertices in the given UDG and k is the size of the maximal independent set.

Proof. The approximation factor and the time complexity result follow from Equation 3.1 and Lemma 3.3.2, respectively. □

Corollary 3.3.1. *The semi-total domination problem achieves a* $\frac{53}{9}$*-factor approximation result in UDGs with running time* $O(n^2)$*, where n is the number of vertices in the given UDG.*

Proof. From Lemma 3.1.3 and Lemma 3.3.3, we have $|D| \leq \frac{44}{9}|D^*|$ and $|T| \leq |D^*|$, respectively. Therefore,

$$\begin{aligned} |D_{t2}| = |D \cup T| &\leq |D| + |T| \\ &\leq \frac{44}{9}|D^*| + |D^*| = \frac{53}{9}|D^*| \\ &\leq \frac{53}{9}|D^*_{t2}| \end{aligned} \tag{3.2}$$

□

3.4 A $2 + \ln(\Delta + 1)$-factor Approximation Algorithm

In [69], the authors proposed a $2+3\ln(\Delta+1)$-factor approximation algorithm for the *semi-total dominating set problem* in general graphs. Here, the authors used two sets, namely D and T, to find the *semi-total dominating set* of the given graph G, where D is a DS and T is a set of vertices such that $D \cup T$ is a *semi-total dominating set*. The approximation algorithm for the *minimum dominating set* (MDS) problem was used to find the set D, and the approximation algorithm for the *minimum set cover* (MSC) problem was used to find the set T. Since the approximation factors for the MDS and MSC problems are $1 + \ln(\Delta + 1)$ and $1 + 2\ln\Delta$, respectively, where Δ represents the degree of the graph. Hence, the overall approximation factor of the algorithm in [69] is $2 + 3\ln(\Delta + 1)$.

However, to improve the approximation factor, we can modify the algorithm in [69] by selecting the set T as done in Algorithm 3.1 (the selection of the set T requires D to be a *dominating set*, not necessarily a *maximal independent set*) . Then, by Lemma 3.3.3 and Observation 3.1.1, the approximation factor of the *semi-total domination problem* in general graphs is as follows:

$$\begin{aligned} |D_{t2}| = |D \cup T| &\leq |D| + |T| \\ &\leq (1 + \ln(\Delta + 1))|D^*| + |D^*| \\ &\leq (2 + \ln(\Delta + 1))|D^*| \\ &\leq (2 + \ln(\Delta + 1))|D^*_{t2}| \end{aligned} \tag{3.3}$$

NOTE: Since there exists a polynomial-time approximation scheme (PTAS) for the domination problem in UDGs with approximation factor $(1 + \epsilon)$ and time $n^{O(\frac{1}{\epsilon}\frac{1}{\log \epsilon})}$, for any $\epsilon > 0$ [91]. We have the following corollary.

Corollary 3.4.1. *The semi-total dominating set problem in UDGs admits a PTAS with approximation factor* $(2 + \epsilon)$ *in time* $n^{O(\frac{1}{\epsilon}\frac{1}{\log \epsilon})}$.

3.5 Conclusion

In this chapter, we have introduced the concept of semi-total domination to UDGs and shown that the *semi-total domination problem* in UDGs is NP-complete. Then, we proposed a 6-factor approximation algorithm for the same, with time complexity $O(n \log k)$, where k is the size of the maximal independent set of the given UDG. Using a similar technique, we improve the approximation factor to $\frac{53}{9}$, but the time complexity of the algorithm is $O(n^2)$. In addition, we also proposed a $2 + \ln(\Delta + 1)$-factor approximation algorithm for general graphs, where Δ is the degree of the given graph. We also observed that the *semi-total domination problem* in UDGs admits a PTAS with approximation factor $(2 + \epsilon)$ in time $n^{O(\frac{1}{\epsilon}\frac{1}{\log \epsilon})}$.

Indian Institute of Technology Guwahati

Chapter 4

Total Domination in UDGs

In this chapter, we explore the *total dominating set* (TDS) problem across various graph subclasses, with a particular focus on UDGs and grid graphs. Consider a set of points $P = \{p_1, p_2, \ldots, p_n\}$, where each point p_i corresponds to the center of a disk D_i for $i = 1, 2, \ldots, n$. A graph $G = (V, E)$ is defined as a geometric UDG if there is a one-to-one correspondence between each $v_i \in V$ and p_i, and there exists an edge $(v_i, v_j) \in E$ if and only if $\delta(c_i, c_j) \leq 1$, where $\delta(c_i, c_j)$ is the Euclidean distance between the two points c_i and c_j in $\mathbb{R}^2$. Given graph $G = (V, E)$, a subset $D_t \subseteq V$ is a *total dominating set* (TDS) of G if (i) D_t is a *dominating set* of G, and (ii) the vertices in D_t induce a subgraph with no isolated vertex. The total dominating set with minimum cardinality is called the *minimum total dominating set* (MTDS), and the size of the *minimum total dominating set* is called the *total domination number*, $\gamma_t(G)$. The geometric version of the TDS problem in UDGs is defined below.

Definition 4. *Given a unit disk graph $G = (V, E)$ corresponding to a point set $P = \{p_1, p_2, \ldots, p_n\}$ in $\mathbb{R}^2$, find a total dominating set of minimum size.*

For smooth referral, we refer the first property of the total domination as *domination property* and the second one as *total property*. As far as the TDS problem in UDGs is concerned,

in 2021, Jena and Das [72] proved that this problem is NP-complete and provided 8-factor approximation algorithm with running time $O(|V| \log |D_t|)$, where $|D_t|$ is the size of the output. They also presented a polynomial time approximation scheme (PTAS) that runs in $O(k^2 n^{2(\lceil 2\sqrt{2}k \rceil)^2})$ time to compute a *total dominating set* of size at most $(1 + \frac{1}{k})^2 |D_t^*|$, where $k \geq 1$ and D_t^* is the minimum TDS. Although this scheme can compute a TDS of size at most $4|D_t^*|$ in $O(n^{18})$ time, the time complexity increases significantly for better approximations. Thus, there is scope for improvements in both the approximation factor and the running time.

To address these challenges, we achieve the following goals:

- develop a 7.79-factor approximation algorithm for the TDS problem in UDGs with time complexity $O(n^2)$, where n is the number of vertices in the given UDG.
- develop a 6.29-factor approximation algorithm for the TDS problem in UDGs with time complexity $O(n^2 m)$, where n and m are the number of vertices and edges in the given UDG.
- prove that the TDS problem is NP-complete when restricted to grid graphs, a subclass of UDGs.

The remainder of the chapter is organized as follows. In Section 4.1, we listed the required preliminaries. In Section 4.2 and Section 4.3, we propose a 7.79-factor and a 6.29-factor approximation algorithms for the TDS problem in UDGs with time complexities $O(n^2)$ and $O(n^2 m)$, respectively. In Section 4.4, we prove the hardness result of the *total dominating set* problem in grid graphs, a subclass of UDGs. We conclude the chapter in Section 4.5.

4.1 Preliminaries

In this section, we define some notations and definitions pertinent to the chapter. For the completeness of the thesis, we revisit some of the already-known facts and properties of the unit disk graphs (UDGs). Here, we also establish some lemmas that are relevant to the chapter.

Let $U(p)$ denote a disk of radius 1 centered at a point $p \in V$ and $U(P)$ denote the unit disks centered at the points in a set $P \subseteq V$, i.e., $U(P) = \{U(p) : p \in P\}$. The set of disks $U(P)$ is said to be independent if for any pair $p, q \in P$, $p \notin U(q)$, i.e., $\delta(p, q) > 1$. We also examine the impact of an additional restriction on grid graphs, where the grid graph is a UDG with all disks having a radius of 1 and centers located exclusively at integer coordinates. In a graph G, we use the symbol $x \overset{G}{\rightsquigarrow} y$ to denote a path between x and y consisting of multiple edges, and we use multiple paths to denote a cycle. For example, the symbol $x \overset{G}{\rightsquigarrow} y \overset{G}{\rightsquigarrow} x$ represents a cycle consisting of paths $x \overset{G}{\rightsquigarrow} y$ and $y \overset{G}{\rightsquigarrow} x$.

Lemma 4.1.1. *[72] Consider two points $p, q \in \mathbb{R}^2$ such that $\delta(p, q) \leq 1$. Let S be the set of independent unit disks such that each disk in S contains the points p and/or q. Then, $|S| \leq 8$.*

Lemma 4.1.2. *Let C be a unit disk centered at c. Suppose $S = \{c_1, c_2, \ldots, c_t\}$ is a set of centers of independent disks, where each center c_i satisfies $1 < \delta(c, c_i) \leq 2$ for $1 \leq i \leq t$. Then, $t \leq 18$.*

Proof. Let $\mathscr{D}$ and D be the disks of radius 2 and 1, respectively, centered at a single point c. We have to show that at most 18 independent unit disks exist whose centers lie on the disk $\mathscr{D}$ and are also independent from the disk D. We prove the result in two steps. In the first step, we show that there exists at most 12 independent unit disks' centers on the periphery of $\mathscr{D}$. Let $S_1 = \{c_1, c_2, \ldots, c_{12}\}$ be the set of corresponding disks' centers such that $\delta(c, c_i) = 2$ for $i = 1, 2, \ldots, 12$. In the second step, we show that if $\delta(c, c_i) = 2$ for $i = 1, 2, \ldots, 12$; then there exists at most 6 independent unit disks, say $S_2 = \{c'_1, c'_2, \ldots, c'_6\}$ such that $U(S)$ is independent where $S = S_1 \cup S_2$ and $1 < \delta(c, c'_i) < 2$ for $i = 1, 2, \ldots, 6$.

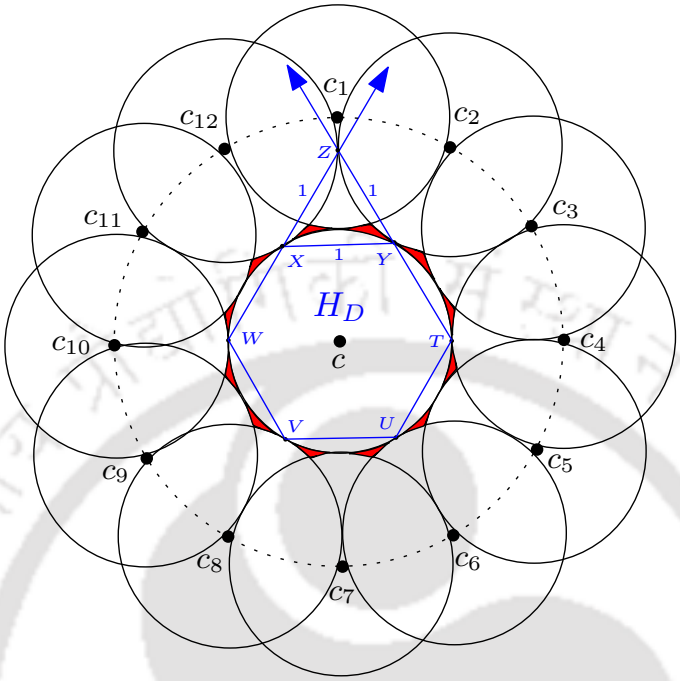

Figure 4.1: *Illustration of independent unit disks inside a disk of radius* 2

(i) On contrary assume that $S_1 = \{c_1, c_2, \ldots, c_{13}\}$ is the set of 13 points such that $\delta(c, c_i) = 2$, for $1 \leq i \leq 13$. Let the points c_1, c_2,..., c_{13} be placed sequentially in clockwise order on the periphery of the disk $\mathscr{D}$ (i.e., $\delta(c, c_i) = 2$ for $1 \leq i \leq 13$). Let $\overline{\mathrm{cc_i}}$ be the line segment joining the point c with c_i, where $1 \leq i \leq 13$. Then there exists at least one pair of consecutive segments $\overline{\mathrm{cc_j}}$ and $\overline{\mathrm{cc_k}}$ such that $\angle c_j c c_k \leq \frac{2\pi}{13}$. Without loss of generality, let the two segments be $\overline{\mathrm{cc_1}}$ and $\overline{\mathrm{cc_2}}$ such that $\angle c_1 c c_2 \leq \frac{2\pi}{13}$. We need to show that the points c_1 and c_2 are not independent. Let's consider the triangle $\triangle c c_1 c_2$ as shown in Figure 4.2. Let $\overline{\mathrm{cp}}$ be the perpendicular bisector of $\overline{\mathrm{c_1c_2}}$. Then $|\overline{\mathrm{c_1c_2}}| = 4\sin(\frac{2\pi}{13\times 2}) < 1$. This leads to the contradiction that c_1 and c_2 are independent. This proves that $|S_1| \leq 12$ (refer to Figure 4.1 for an orientation of 12 independent disks whose centers are placed on the periphery (dotted line) of the disk $\mathscr{D}$).

(ii) From the first step, it is proved that there can be at most 12 independent unit disks whose centers lie on the periphery of a disk of radius 2. Let $S_1 = \{c_1, c_2, \ldots, c_{12}\}$ be the set of disks' centers that lie on the periphery of $\mathscr{D}$ as shown in Figure 4.1. Let D be a unit disk

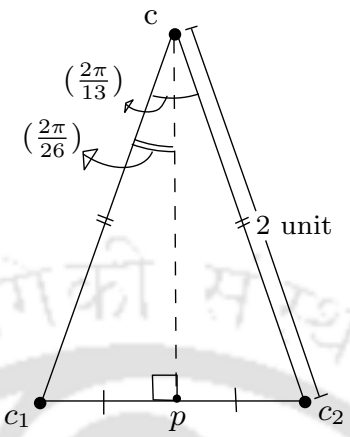

Figure 4.2: *Measuring distance of* $\overline{c_1c_2}$

centered at point c; then there exists a set of 12 regions, say $R = \mathscr{D} \setminus D \setminus \bigcup_{i=1}^{12} U(c_i)$ (the 12 regions are shaded in Red colour in Figure 4.1, which are still independent from S_1. Now, we have to show that at most 6 independent unit disk centers can lie in these regions. Let H_D $(TUVWXY)$ be the hexagon inscribed in disk D as shown in Figure 4.1. Since D is a unit disk, each arm of the hexagon H_D is of unit length. Let us extend the line segments $\overline{WX}$ and $\overline{TY}$ such that they meet at point Z. Now, the $\triangle XYZ$ is an equilateral triangle with side length of 1 unit. Since each such triangle contains two shaded (red) regions, no consecutive regions can contain the centers of two independent unit disks (since the maximum distance between any two points in an equilateral triangle of side length 1 is upper bounded by 1). Therefore, 12 regions can contain at most 6 independent unit disks. Let $S_2 = \{c'_1, c'_2, \ldots, c'_6\}$ be the independent unit disks on those 12 regions.

From the first step, we know that there exists at most 12 independent unit disks at a distance 2 from the point c. If these disks move closer to c, their independence decreases, which also reduces the combined area of the 12 regions. Therefore, placing the centers of the 12 disks on the periphery ensures the maximum combined area, resulting in $|S| = |S_1 \cup S_2| \leq 18$. □

Observation 4.1.1. *[44] Let G be a graph with no isolated vertex. Then, $\gamma(G) \leq \gamma_t(G)$.*

Minimum Set Cover (MSC) problem: The *Minimum Set Cover* (MSC) problem involves a universal set $\mathbb{U} = \{u_1, u_2, \ldots, u_n\}$ and a set of subsets $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_m\}$, where

each $\mathbb{S}_i$ is a subset of $\mathbb{U}$ and $\mathbb{U} = \bigcup_{i=1}^{m} \mathbb{S}_i$. The goal of the MSC problem is to find the smallest subset $\mathbb{T} \subseteq \mathbb{S}$ such that the union of all subsets in $\mathbb{T}$ covers $\mathbb{U}$. We denote an instance of the MSC problem as $< \mathbb{U}, \mathbb{S} >$, where $\mathbb{U}$ represents a finite set known as the universal set, and $\mathbb{S}$ constitutes a family of subsets of $\mathbb{U}$.

Theorem 4.1.1. *[30] The MSC problem can be approximated with an approximation factor* $H(max\{|\mathbb{S}_i| : \mathbb{S}_i \in \mathbb{S}\})$ *using* $GreedySetCover(\mathbb{U}, \mathbb{S})$ *in time* $O(n^2m)$*, where* $H(m)$ *is the* m*-th harmonic number.*

4.2 A 7.79-factor Approximation Algorithm

In this section, we propose an algorithm called *TDS-UDG* (see Algorithm 4.1 for the pseudocode) for the TDS problem in geometric UDGs with an approximation factor 7.79. The algorithm runs on a graph with no isolated vertex. If the graph is disconnected, each component can run it to obtain the TDS.

4.2.1 Algorithm

We briefly describe the algorithm for finding a *total dominating set* (TDS) D_t of the given geometric UDG G. *TDS-UDG* algorithm consists of three phases. The first phase segregates the set of vertices V into $k+1$ subsets, i.e., $S = \{S_i : 0 \leq i \leq k\}$ such that $V = \bigcup_{i=0}^{k} S_i$. The second phase selects a set of vertices I from S_i (where $0 \leq i \leq k$) such that I satisfies the *domination property* of graph G. At the end, the third phase identifies another set of vertices $T \subseteq V$ that satisfies the *total property*. Finally, *TDS-UDG* reports the *total dominating set* D_t as the collection of the two sets I and T, i.e., $D_t = I \cup T$. Now, we discuss each of the phases of the algorithm elaborately.

4.2.1.1 Phase I

In phase I, the *TDS-UDG* algorithm divides the vertex set V into $k+1$ subsets named S_0, S_1,..., S_k. This segregation is achieved by constructing a *breadth-first search* (BFS) tree

Algorithm 4.1 TDS-UDG

Require: A unit disk graph $G = (V, E)$ with known disk centers
Ensure: A TDS D_t

Choose an arbitrary vertex $x \in V$
$x.explored = true$, $x.parent = NIL$
for each vertex $u \in V \setminus \{x\}$ **do**
    $u.explored = false$, $u.parent = NIL$
end for
$I = \emptyset$, $T = \emptyset$, $D_t = \emptyset$, $k = 0$, $count = 0$, $S_k = \{x\}$, $I_k = \{x\}$
while $count \neq |V|$ **do**
    $S_{k+1} = \emptyset$
    for each node $u \in S_k$ **do**
        for each node $v \in N_G(u)$ **do**
            if $v.explored = false$ **then**
                $v.explored = true$, $S_{k+1} = S_{k+1} \cup \{v\}$
                $v.parent = u$
            end if
        end for
        $count = count + 1$
    end for
    $k = k + 1$
end while
$k = k - 1$
if $k = 1$ **then**
    choose any vertex $y \in N(x)$
    $T = T \cup \{y\}$
else
    for $i = 2$ to k **do**
        $I_i = \emptyset$, $T_i = \emptyset$
        while $S_i \neq \emptyset$ **do**
            Choose a vertex $p \in S_i$
            $I_i = I_i \cup \{p\}$
            $T_i = T_i \cup \{p.parent\}$
            $S_i = S_i \setminus N_G[p]$, $S_{i+1} = S_{i+1} \setminus N_G[p]$
        end while
        $I = I \cup I_i$
        $T = T \cup T_i$
    end for
end if
$D_t = I \cup T$
return D_t

rooted at vertex x. Let T_x denote the BFS tree rooted at x for graph G (see Figure 4.3 and the corresponding rooted tree T_x in Figure 4.4). In this tree T_x, if a vertex y is at level i, then $y \in S_i$, where x is considered to be at level 0. The level of a vertex in T_x refers to the number of edges in the path connecting it to the root x. Therefore, S_i can be interpreted as the collection of all the vertices that are at i^{th} level in the tree T_x (see Lines 1-19 of Algorithm 4.1). This construction ensures that each subset S_i corresponds to vertices at increasing distances from x in T_x, facilitating a structured way to partition V based on BFS levels.

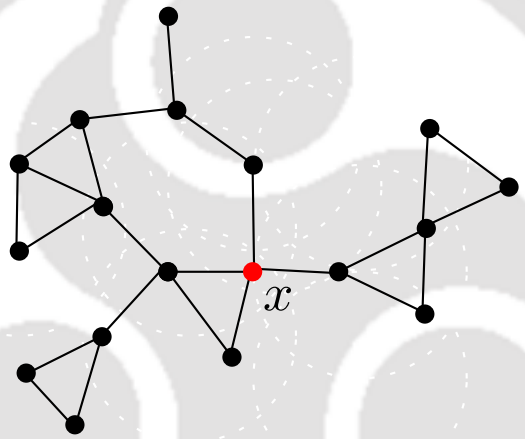

Figure 4.3: $G = (V, E)$

Observation 4.2.1. *Every vertex $v \in S_i$ has a neighbor in S_{i-1} for $i = 1, 2, \ldots k$.*

Proof. In T_x, i represents the level of a node, and S_i is the collection of nodes with level i. Since T_x is connected, every vertex $v \in S_i$ has a neighbor in S_{i-1}. □

Observation 4.2.2. *If T_x is the rooted BFS tree constructed at vertex x of G, then for any vertex $u \in S_i$, there does not exist any vertex $v \in S_i$ such that $uv \in E(T_x)$, where $0 \leq i \leq k$ and $E(T_x)$ is the set of edges in T_x.*

Proof. On the contrary, assume that there exist two vertices $u, v \in S_i$ such that $uv \in E(T_x)$. Since the vertices u and v are in S_i (i.e., at level i in T_x), there exists paths $u \overset{T_x}{\rightsquigarrow} x$ and

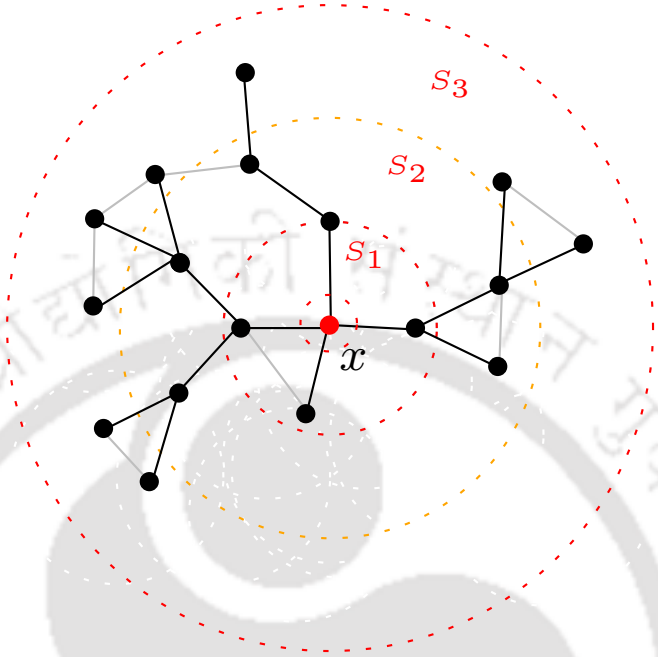

Figure 4.4: *Rooted BFS tree T_x at x of G*

$v \overset{T_x}{\rightsquigarrow} x$ in T_x. Now, if $uv \in E(T_x)$, then the edge uv forms a cycle $x \overset{T_x}{\rightsquigarrow} u \overset{T_x}{\rightsquigarrow} v \overset{T_x}{\rightsquigarrow} x$ in T_x. This leads to a contradiction that T_x is a tree. □

4.2.1.2 Phase II

In phase II, the *TDS-UDG* algorithm constructs a set of independent vertices I_i for each S_i such that $I = \bigcup_{i=0}^{k} I_i$ is a maximal independent set of G. Initially, I_0 contains x as S_0 contains the only vertex x (see Line 6 in Algorithm 4.1). Since each vertex in S_1 is adjacent to x, $I_1 = \emptyset$. For the remaining S_i for $2 \leq i \leq k$, the set I_i is constructed on an incremental basis, i.e., for $2 \leq i \leq k$, I_i is incrementally constructed after I_{i-1}. The algorithm randomly selects a vertex $p \in S_i$ as a candidate for I_i. It then deletes the closed neighborhood of p in G (i.e., $N_G[p]$) from S_i and S_{i+1} (see Line 31 in Algorithm 4.1). This deletion ensures that subsequent candidates for I_i remain independent from the vertices already in I_i, maintaining $I_i \cup I_{i+1}$ as an independent set. This process continues until S_i is exhausted. When i reaches $k+1$, I becomes a maximal independent set of G (see Line 33 in Algorithm 4.1). Since every maximal independent set of a graph is a *dominating set* of

the graph, the maximal independent set I obtained by the algorithm is a *dominating set* of G, satisfying the *domination property*.

4.2.1.3 Phase III

In phase III, the *TDS-UDG* algorithm finds a set T that satisfies the *total property* when taken with the independent set I. Here, $T = \bigcup_{i=0}^{k} T_i$, where T_i represents the parent node corresponding to each vertex selected in I_i (see Line 30 in Algorithm 4.1). The algorithm maintains a record of the parent vertex for each node present in S_i (as indicated by Line 13). It updates the set T by adding the parent nodes associated with each vertex selected in I (see Lines 29-30 in Algorithm 4.1). By combining I with T, the algorithm ensures that the vertices in $G[I \cup T]$ remain free from any isolated vertex. This guarantees that the constructed set I along with T maintains the *total property*.

Lemma 4.2.1. *Algorithm 4.1 returns (D_t) a TDS of the geometric UDG G.*

Proof. Since $D_t = I \cup T$ and from Phase I, Phase II, and Phase III, it is clear that the set D_t satisfies both the domination and the total properties. Hence, the lemma follows. □

Lemma 4.2.2. *Algorithm 4.1 runs in $O(|V| + |E|)$ time.*

Proof. The time complexity of the algorithm *TDS-UDG* (Algorithm 4.1) is primarily dominated by the *nested loop* used for segregating the set V into k subsets (see Lines 7-19 in Algorithm 4.1). In the worst case, the algorithm checks each vertex and the corresponding adjacency list for segregation. Hence, the time complexity of the algorithm *TDS-UDG* is $O(|V| + |E|)$. □

Lemma 4.2.3. *In Algorithm 4.1, if $|I| \geq 2$, then for each vertex $u \in I$, there exists at least one vertex $v \in I$, such that $\delta(u, v) \leq 2$.*

Proof. Since Algorithm 4.1 constructs a maximal independent set of G on an incremental basis, i.e., it constructs the maximal independent set I_i of S_i only after constructing the maximal independent set I_{i-1} of S_{i-1}, where $I = \bigcup_{i=0}^{k} I_i$. So the lemma can be proved using

induction on k with respect to the rooted tree T_x at x, i.e., the algorithm finds the maximal independent set of the induced subgraph $G[S_0 \cup S_1 \cup \cdots \cup S_i]$ from $G[S_0 \cup S_1 \cup \cdots \cup S_{i-1}]$ to find a maximal independent set of G since $G = G[S_0 \cup S_1 \cup \cdots \cup S_k]$. Let $P(k)$ be the proposition that for each vertex $u \in \bigcup_{i=0}^{k} I_i$, there exists at least one vertex $v \in \bigcup_{i=0}^{k} I_i$ such that $\delta(u, v) \leq 2$ for $k \geq 2$.

Basis step: We have to show that $P(2)$ is true, i.e., when $k = 2$, for each $u \in \bigcup_{i=0}^{2} I_i$, there exists at least one vertex $v \in \bigcup_{i=0}^{2} I_i$ such that $\delta(u, v) \leq 2$. Since, $I_0 = \{x\}$ and $I_1 = \emptyset$ ($S_1 = \emptyset$, since *TDS-UDG* removes $N_G(x)$ from S_1). Let I_2 be the maximal independent set of S_2. Since the graph is connected, each vertex $v \in I_2$ has at least one vertex w in S_1 (see Observation 4.2.1). This implies $\delta(v, w) \leq 1$. Since every vertex present in S_1 is a neighbor of x, $w \in N_G(x)$. So $\delta(x, w) \leq 1$. From the triangle inequality, we have $\delta(v, x) \leq \delta(v, w) + \delta(w, x) \leq 2$. Hence $P(2)$ is true.

Inductive Hypothesis: When $k = m$, $P(m)$ is true, i.e., for each $u \in \bigcup_{i=0}^{m} I_i$, there exists at least one vertex $v \in \bigcup_{i=0}^{m} I_i$ such that $\delta(u, v) \leq 2$.

Inductive Step: We have to show that when $k = m + 1$, $P(m + 1)$ is true, i.e., for each $u \in \bigcup_{i=0}^{m+1} I_i$, there exists at least one vertex $v \in \bigcup_{i=0}^{m+1} I_i$ such that $\delta(u, v) \leq 2$. Since $P(m)$ is true, the following is sufficient to prove the lemma: for each vertex $u \in I_{m+1}$; there exists at least one vertex $v \in I_{m-1}$ such that $\delta(u, v) \leq 2$.

Without loss of generality, let us consider a vertex $u \in I_{m+1}$. Then the vertex u has one neighbor $w \in S_m$ (see Observation 4.2.1). Since $uw \in E$, $\delta(u, w) \leq 1$. $u \in I_{m+1}$ implies $u \in S_{m+1}$. u is still in S_{m+1} because none of the neighbors of u is in I_m; otherwise, u would have been removed from S_{m+1} and hence u should not have appeared as a candidate in I_{m+1}. Since $P(m)$ is true and $w \notin I_m$, there exists a vertex $v \in I_{m-1}$ due to which w was removed from S_m. This implies $\delta(w, v) \leq 1$. Due to triangle inequality $\delta(u, v) \leq \delta(u, w) + \delta(w, v) \leq 2$.

Hence, the proposition holds. □

Lemma 4.2.4. *In the worst case,* 19 *vertices in* I *share* 18 *nodes of* T *as intermediate nodes.*

Proof. Since I is an independent set and for each node $u \in I$, another node $v \in I$ exists, such as $\delta(u, v) \leq 2$ (see Lemma 4.2.3). From Lemma 4.1.2, we know that a disk of radius 2 contains at most 19 independent unit disks. So, in the worst case, those 19 vertices can share 18 nodes of T as intermediate nodes. □

Observation 4.2.3. *If* $k = 1$*, then* $|D_t| = |D_t^*|$*, where* D_t *is the TDS returned by algorithm TDS-UDG and* D_t^* *is the optimal TDS of* G*.*

Proof. If $k = 1$, the rooted tree T_x is a star graph with center vertex x. Hence $|D_t| = |D_t^*| = 2$. □

4.2.2 Analysis

Let D_t^* be the optimal *total dominating set* of the graph G. The set D_t in *TDS-UDG* is a TDS of G, where $D_t = I \cup T$ (see Lemma 4.2.1). Since D_t^* is a TDS of G, there is no isolated vertex in $G[D_t^*]$. For each $pq \in E(G[D_t^*])$, there exists at most 8 independent vertices in I which contains the vertices p and/or q (see Lemma 4.1.1). Since the 2 vertices of an edge in $G[D_t^*]$ dominate 8 vertices of I, the following equation holds.

$$|I| \leq 4|D_t^*| \tag{4.1}$$

The set T in *TDS-UDG* satisfies the *total property* when added to the independent set I. In the worst case, we know that 19 vertices in I share 18 vertices from T as intermediate nodes (see Lemma 4.2.4). So there are at most $\frac{18}{19} \times 4|D_t^*|$ vertices in T for $4|D_t^*|$ number of vertices in I, i.e.,

$$|T| \leq \frac{72}{19}|D_t^*| \tag{4.2}$$

Therefore, from Lemma 4.2.1,

$$\begin{aligned} |D_t| &= |I| + |T| \\ &\leq 4|D_t^*| + \frac{72}{19}|D_t^*| \\ &= \frac{148}{19}|D_t^*| \\ &\leq 7.79|D_t^*| \end{aligned} \tag{4.3}$$

Theorem 4.2.1. *The proposed algorithm (TDS-UDG) gives a 7.79-factor approximation algorithm for the TDS problem in UDGs, which runs in $O(|V| + |E|)$ time.*

Proof. The approximation factor follows from Equation 4.3, and the time complexity result follows from Lemma 4.2.2. □

4.3 A 6.29-factor Approximation Algorithm

In this section, we propose a 6.29-factor approximation algorithm for the TDS problem in UDGs.

4.3.1 Algorithm

The algorithm aims to find a TDS in a unit disk graph $G = (V, E)$, where $V = \{p_1, p_2, \ldots, p_n\}$ represents the centers of unit disks in $\mathbb{R}^2$. To begin, the algorithm identifies a maximal independent set $D \subseteq V$, which satisfies the *domination property* (see Lines 2-6 of Algorithm 4.2). Next, to fulfill the *total property*, the algorithm selects a subset $T \subseteq V$ such that for each $v \in D$, there exists a vertex $u \in V \setminus D$ where u is adjacent to v. To find the set T, the algorithm constructs a set cover instance $< D, S >$, where D serves as the universal set, and S consists of subsets $S_i = N_G(u_i) \cap D$ for each $u_i \in V \setminus D$ (see Lines 8-12 of Algorithm 4.2). Here, $N_G(u_i) \cap D$ represents the neighbors of u_i within D. Subsequently, a greedy set cover algorithm is applied to $< D, S >$ to find a minimal subset $S' \subseteq S$ such that $\bigcup_{S_i \in S'} S_i = D$ (see Line 13 of Algorithm 4.2). For each $S_i \in S'$, T contains the corresponding u_i of $V \setminus D$

(see Line 14 of Algorithm 4.2). Then, it reports $D_t = D \cup T$ as a TDS of G. A complete illustration of the scenario is shown by an example in Figure 4.6(a), in which the edges between the set D and $V \setminus D$ are only shown. Lemma 4.3.2 and Lemma 4.3.3 represent the algorithm's correctness and time complexity, respectively.

Algorithm 4.2 TDS-UDG-SC

Input: A unit disk graph, $G = (V, E)$, with known disk centers
Output: A TDS D_t for G

1: $V' = V, D = \emptyset, S = \emptyset$
2: **while** $V' \neq \emptyset$ **do** ▷ *domination* property of TDS
3: choose a vertex $v \in V'$
4: $D = D \cup \{v\}$
5: $V' = V' \setminus N_G[v]$
6: **end while**
7: $i = 1$
8: **for** each $u \in V \setminus D$ **do**
9: $S_i = N_G(u) \cap D$
10: $i = i + 1$
11: $S = S \cup S_i$
12: **end for**
13: $S' = GreedySetCover(D, S)$
14: $T = \{u_i | S_i \in S'\}$ ▷ *total* property of TDS
15: $D_t = D \cup T$
16: **return** $|D_t|$

Lemma 4.3.1. *Given a maximal independent set D of a geometric unit disk graph $G = (V, E)$ such that $S = \{S_i\}$, where $S_i = D \cap N_G(u_i)$, for each $u_i \in V \setminus D$. If C^* is an optimal* set cover *of the set cover instance $< D, S >$ and D^* is an optimal dominating set of G, then $|C^*| \leq |D^*|$.*

Proof. Given that D is a *maximal independent set* in the unit disk graph G, for any vertex $v \in V \setminus D$, the Lemma 3.1.2 ensures that $|N_G(v) \cap D| \leq 5$. Let D^* denote an *optimal dominating set* of G, then either (i) $D^* \subseteq D$ or (ii) $D^* \subseteq V \setminus D$ or (iii) $D^* = D^*_D \cup D^*_{V \setminus D}$, where $D^*_D = D^* \cap D$ and $D^*_{V \setminus D} = D^* \cap (V \setminus D)$. We need to demonstrate that for each of the given scenarios $|C^*| \leq |D^*|$.

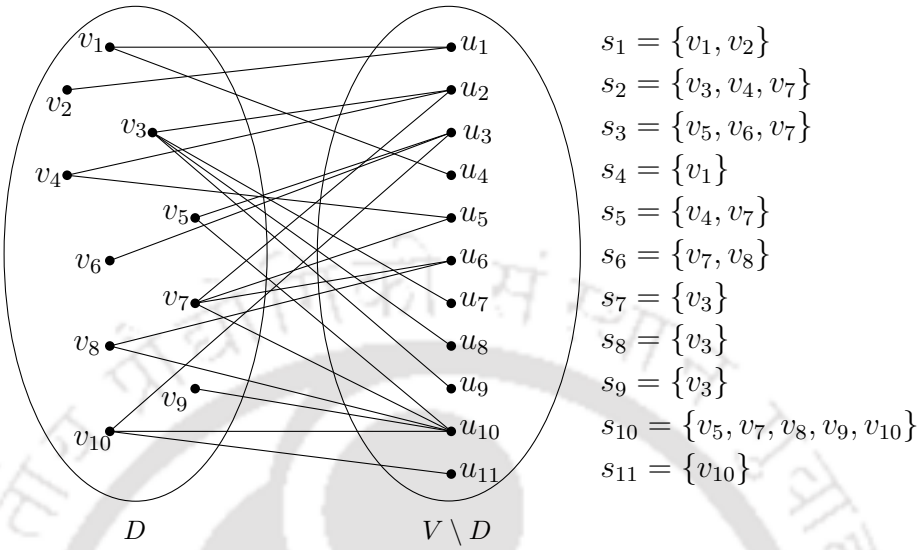

Figure 4.5: *Illustration of D*

(i) $D^* \subseteq D$: If $D^* \subseteq D$, then $D^* = D$; otherwise, there exists at least one vertex $v \in D$, such that v is not dominated by D^* (since D is an *independent set*). This leads to a contradiction that D^* is a dominating set. Since D is an independent set, in the worst case, D requires at most $|D|$ number of subsets (i.e., vertices) from S (i.e., $V \setminus D$) to cover D. If C^* is an optimal cover of the set cover instance $< D, S >$, then $|C^*| \leq |D| = |D^*|$.

(ii) $D^* \subseteq V \setminus D$: On the contrary, let us assume $C^* > D^*$. Since D^* is a dominating set, each vertex in $D \cup V \setminus D$ (i.e., V) is either in D^* or is adjacent to at least one vertex in D^*. If so, a set $D' \subseteq D^*$ exists such that D' dominates D (since D is an independent set). This implies $|D'|$ number of subsets (i.e., vertices) from S (i.e., $V \setminus D$) are sufficient to cover D, which is less than or equal to $|C^*|$. This leads to a contradiction to the fact that C^* is an optimal cover of D.

(iii) $D^* = D^*_D \cup D^*_{V \setminus D}$, where $D^*_D = D^* \cap D$ and $D^*_{V \setminus D} = D^* \cap (V \setminus D)$ (refer to Figure 4.6 for a pictorial representation): Since $D^*_D \subseteq D$, D^*_D is an independent set. In the worst case, at most $|D^*_D|$ number of subsets (vertices) from S are required to cover vertices in D^*_D. Since $D^*_{V \setminus D}$ dominates the set $D \setminus D^*_D$, $|D^*_{V \setminus D}|$ number of subsets (vertices) are sufficient for the coverage of remaining vertices in $D \setminus D^*_D$. Therefore, $|C^*| \leq |D^*_D| + |D^*_{V \setminus D}| = |D^*|$. □

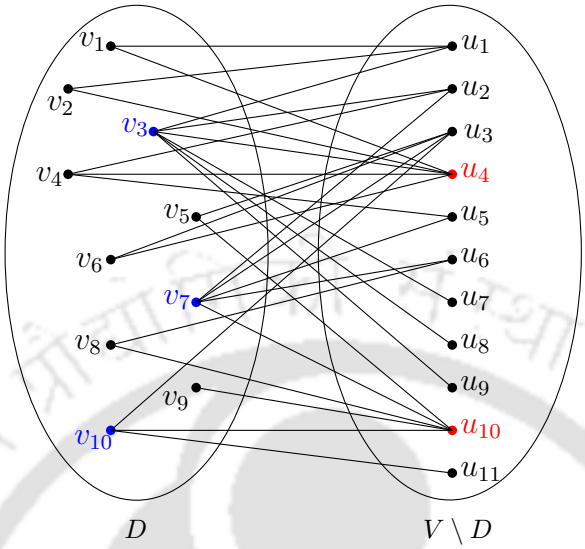

Figure 4.6: *Illustration of $V \setminus D$*

Lemma 4.3.2. *The set D_t in Algorithm 4.2 is a TDS of G.*

Proof. In the first phase, Algorithm 4.2 finds a *maximal independent set* D of G to satisfy the domination property (see Lines 2-6 in Algorithm 4.2). Next, the algorithm runs $GreedySetCover(D, S)$ to find a subset T such that $T = \{u_i | S_i \in S'\}$ (see Line 13 and Line 14 in Algorithm 4.2). The set T ensures that for each vertex $v \in D$, there exists a vertex $u \in T$ such that $uv \in E$. Hence, the set T, when combined with D, confirms that none of the vertices in D is isolated. Therefore, the nominated points in D and T satisfy the *domination* and *total* properties. Consequently, the set D_t forms a TDS of G. □

Lemma 4.3.3. *Algorithm 4.2 runs in $O(n^2m)$ time, where n and m are the number of vertices and edges of G, respectively.*

Proof. The time complexity of Algorithm 4.1 is primarily dominated by the time taken by the function $GreedySetCover()$. Since $GreedySetCover()$ requires $O(n^2m)$ time (see Theorem 4.1.1), the running time of the Algorithm 4.2 is $O(n^2m)$. □

4.3.2 Analysis

The set D_t in *TDS-UDG-SC* is a TDS of G, where $D_t = D \cup T$ (see Lemma 4.3.2). Let D^* and D_t^* be an optimal DS and an optimal TDS of G, respectively. Since 2 vertices of

an edge in $G[D_t^*]$ dominate at most 8 vertices of D (Lemma 4.1.1), the following equation holds.

$$|D| \leq 4|D_t^*| \tag{4.4}$$

The set T in *TDS-UDG-SC* satisfies the total property when added to the independent set D, where $T = \{u_i | S_i \in S'\}$ and S' is the set cover of the instance $< D, S >$. Let C^* be an *optimal set cover* of $< D, S >$. Then from Lemma 4.3.1, the following equation holds.

$$|C^*| \leq |D^*| \tag{4.5}$$

Therefore, from Lemma 4.3.2, we conclude the following:

$$\begin{aligned} |D_t| &= |D| + |T| \\ &\leq 4|D_t^*| + H(5) \times |C^*| \text{ (refer to Equation 4.4 and Theorem 4.1.1)} \\ &\leq 4|D_t^*| + \frac{137}{60} \times |D^*| \text{ (refer to Lemma 4.3.1)} \\ &\leq 4|D_t^*| + \frac{137}{60} \times |D_t^*| \text{ (refer to Observation 4.1.1)} \\ &= \frac{377}{60} \times |D_t^*| \\ &\leq 6.29 \times |D_t^*| \end{aligned} \tag{4.6}$$

Theorem 4.3.1. *Algorithm 4.2 (TDS-UDG-SC) gives a 6.29-factor approximation result for the TDS problem in UDGs. The algorithm runs in $O(n^2m)$ time, where n and m are the number of vertices and edges in G, respectively.*

Proof. The approximation factor and the time complexity result follow from Equation 4.6 and Lemma 4.3.3, respectively. □

4.4 NP-completeness Result

In this section, we aim to establish the TDS problem's hardness result by demonstrating that its decision version is NP-complete in grid graphs. To achieve this, we employ a reduc-

tion from the decision version of the *vertex cover* (VC) problem in planar graphs with degree at most 3 to the decision version of the TDS problem in grid graphs. Since the reduction is based on geometry, we often refer to a node or vertex as a 'point'. Formally, the decision versions of the above stated problems are as follows:

Decision version of the VC problem in planar graphs of degree at most 3 **(**$D\text{-}VC\text{-}PGD3$**):** Given a planar graph G of degree at most 3 and a positive integer K, does G have a VC of size at most K?

Decision version of the TDS problem in grid graphs ($D\text{-}TDS\text{-}GGs$**):** Given a grid graph G and a positive integer K, does G have a TDS of size at most K?

We prove the hardness result of the TDS problem in grid graphs by making a polynomial time reduction from an arbitrary instance of **D-VC-PGD3** to an instance of **D-TDS-GGs**. To demonstrate this assertion, we use the following lemma (Lemma 4.4.1) given by Valiant.

Lemma 4.4.1. *[103] Let $G = (V, E)$ be a planar graph of degree at most* 3*. The graph G can be embedded in a grid of area $O(|V|^2)$ such that each $v \in V$ is positioned at a grid point with coordinates $(10i, 10j)$, where i and j are integers, and each edge $e \in E$ is a finite sequence of consecutive segments, each of length 10 units, aligned along the grid lines.*

Lemma 4.4.2. *If $G = (V, E)$ is an instance of **D-VC-PGD3** without any isolated vertex, then an instance $G' = (V', E')$ of **D-TDS-GGs** can be constructed from G in polynomial time.*

Proof. Let $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$ be the vertex set and edge set of the instance G. We construct a graph $G' = (V', E')$ from G by the four steps as given below:

Step 1 **(Embedding):** The graph G (refer Figure 4.7(a)) is initially embedded onto a grid with cells of size 10×10 (refer Figure 4.8), utilizing Valiant's algorithm (Lemma 4.4.1)

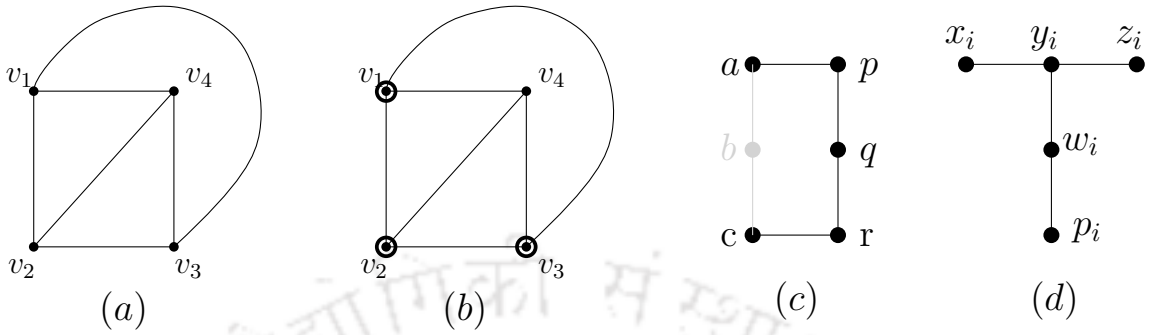

Figure 4.7: *(a) $G = (V, E)$, (b) $S_{vc} = \{v_1, v_2, v_3\}$, vertex cover of G, (c) path, and (d) gadget*

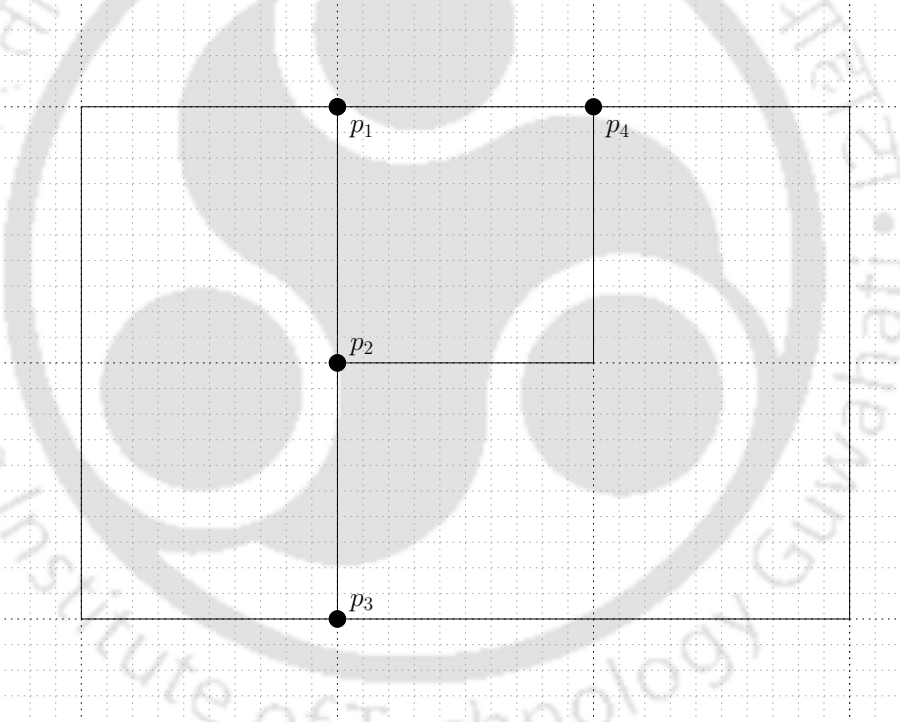

Figure 4.8: *Embedding of G on a grid of cell size* 10×10

as outlined in [103]. Each edge $e \in E$ is represented as a consecutive sequence of line segments on the grid, where each segment has a length of 10 units. Let ℓ_{ij} denote the total number of line segments in $p_i p_j$. For every vertex $v_i \in V$, a **node point** is positioned at a grid coordinate of the form $(10i, 10j)$, where i and j are integers. Let p_i represent the node point located at the grid corresponding to vertex $v_i \in V$, where $1 \leq i \leq n$. We refer these node points as embedding nodes. Let N be the set of these nodes and $|N| = |V| = n$.

Step 2 **(Inclusion of auxiliary nodes):** In this step, we augment each line segment p_ip_j with additional auxiliary nodes. Given that each p_ip_j consists of ℓ_{ij} segments, there will be $10\ell_{ij} - 1$ intermediary grid points along p_ip_j (when the cell size 1×1), excluding p_i and p_j. Consequently, a node is added at each grid point, as illustrated in Figure 4.9. A modification is made if the count of nodes added along each p_ip_j does not conform to the pattern $4k_{ij} + 1$, where k_{ij} is an integer. Specifically, three consecutive grid points, denoted as a, b, and c (excluding those adjacent to either p_i or p_j), are selected. Subsequently, point b and its incident edges are removed, and three new nodes, p, q, and r, are introduced to p_ip_j such that the path $apqrc$ is formed (refer to Figure 4.7(c) and Figure 4.9). This adjustment ensures that the count of grid points on p_ip_j becomes $4k_{ij} + 1$, where k_{ij} is an integer. Let A be the set representing the auxiliary nodes added in this step. $|A| = \sum_{v_iv_j \in E}(4k_{ij} + 1) = 4\sum_{v_iv_j \in E} k_{ij} + |E| = 4\sum_{v_iv_j \in E} k_{ij} + m$.

Step 3 **(Inclusion of gadgets):** In this stage, considering that each node within the planar graph has a degree at most 3 and is embedded within a grid, there exists at least one feasible position at every embedded node to accommodate an additional edge. Consequently, we introduce a special construction called gadget (refer to Figure 4.7(d)), at each embedded node p_i. This gadget at p_i comprises four distinct nodes: x_i, y_i, z_i, and w_i. Let S represent the set of nodes added in this process. Given that each gadget comprises four nodes (excluding node point), it follows that $|S| = 4|N| = 4n$.

Step 4 **(Construction of the grid graph):** Let $G' = (V', E')$ be the grid graph constructed after applying the above 3 steps on graph G, where $V' = N \cup A \cup S$ and $E' = \{uv : u, v \in V' \text{ and } \delta(u, v) = 1\}$, where $\delta(u, v)$ is the Euclidean distance between u and v.

From Lemma 4.4.1, we conclude that the number of segments $\ell = \sum_{v_iv_j \in E} \ell_{ij} = O(n^2)$. Therefore, the upper bound on the number of vertices and the number of edges in G' is

$O(n^2)$. Hence, G' can be constructed from G in polynomial time. □

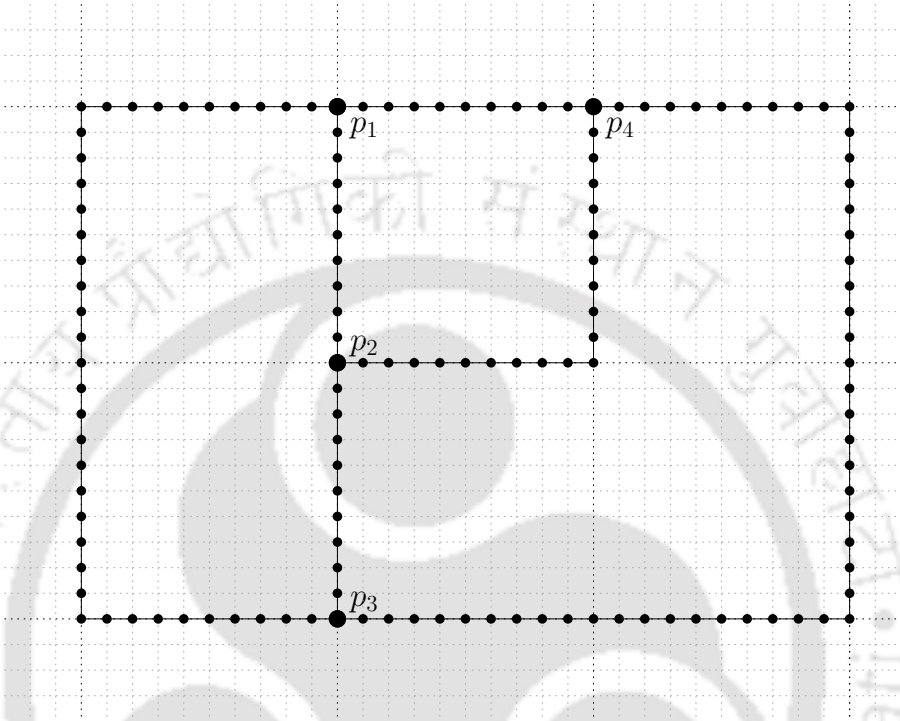

Figure 4.9: *Inclusion of auxiliary points*

Theorem 4.4.1. ***D-TDS-GGs*** *belongs to the class NP-complete.*

Proof. Let $G = (V, E)$ be a grid graph. Given a subset $S \subseteq V$ and a positive integer k, we can verify whether S is a TDS of G of size at most k or not in polynomial time. Therefore, $D\text{-}TDS\text{-}GGs \in NP$.

To establish the NP-hardness of $D\text{-}TDS\text{-}GGs$, we employ a polynomial time reduction from *D-VC-PGD3* to *D-TDS-GGs*. Utilizing Lemma 4.4.2, we can construct an instance $G' = (V', E')$ of *D-TDS-GGs* from an arbitrary instance $G = (V, E)$ of *D-VC-PGD3* in polynomial time. From Claim 4.4.2.1, we have $D\text{-}TDS\text{-}GGs \in NP\text{-}hard$. Therefore, $D\text{-}TDS\text{-}GGs \in NP\text{-}complete$. □

Claim 4.4.2.1. *G has a vertex cover S_{vc} of size at most K if and only if G' has a total dominating set D_t of size at most $K + 2n + 2\sum_{v_iv_j \in E} k_{ij}$ for some integer k_{ij}.*

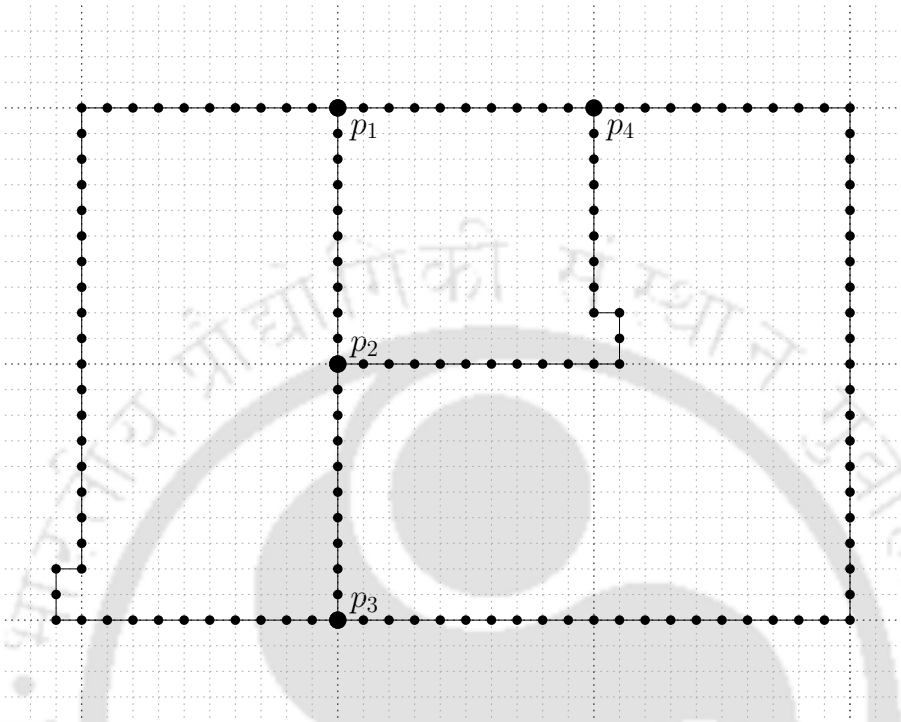

Figure 4.10: *Inclusion of an additional path if needed*

Proof. ($\Longrightarrow$) Let S_{vc} be a *vertex cover* of G such that $|S_{vc}| \leq K$, and let T_{vc} be the set of vertices in G' corresponding to the vertices in S_{vc}, i.e., $T_{vc} = \{p_i \in V' : v_i \in S_{vc}\}$. Now, we construct two sets $T_a \subseteq A$ and $T_g \subseteq S$ such that $D_t = T_{vc} \cup T_a \cup T_g$ forms a *total dominating set* with cardinality less than or equal to $K + 2n + 2\sum_{v_iv_j \in E} k_{ij}$ for some integer k_{ij}.

The construction of T_a and T_g proceeds as follows: since S_{vc} is a *vertex cover* of G (see Figure 4.7(b)), at least one endpoint of every edge in G is inside S_{vc}. Since every edge in G corresponds to a sequence of segments in G', we start from the endpoint inside T_{vc}. For each p_ip_j in G' corresponding to each $v_iv_j \in E$, at least one out of p_i and p_j is in T_{vc}. Without loss of generality, let $p_i \in T_{vc}$ (if $p_i, p_j \in T_{vc}$, then break the tie by picking anyone). We traverse from p_i towards p_j. While traversing, we leave two vertices next to p_i, add the next two vertices to T_a, then leave the next two vertices and select the next two vertices for T_a again. This process is repeated until we reach p_j. For each $v_iv_j \in E$, we apply this process to the corresponding p_ip_j in G'. Since p_ip_j contains $4k_{ij} + 1$ number of intermediate

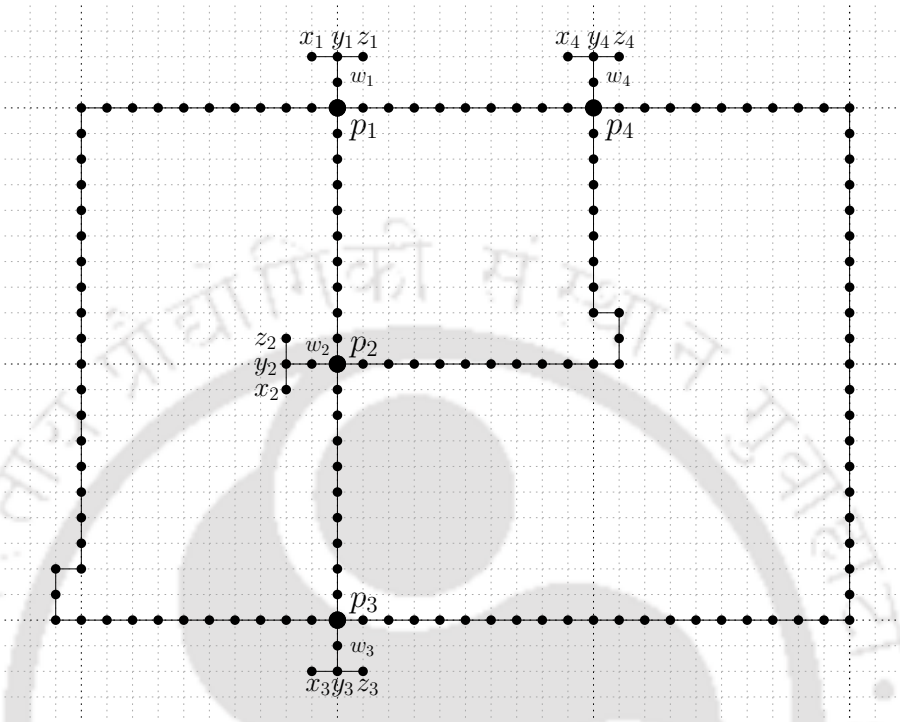

Figure 4.11: *Inclusion of gadget at each node point*

grid points, it requires at least $2k_{ij}$ points from p_ip_j in T_a for total domination. Therefore, $|T_a| = 2\sum_{v_iv_j \in E} k_{ij}$, where k_{ij} is the integer associated to each $v_iv_j \in E$ derived in Step 2 in the proof of Lemma 4.4.2. In Figure 4.12, the red cross points represent T_a.

From each $p_i \in V'$, we choose y_i and w_i for T_g. Hence, $|T_g| = 2n$. In Figure 4.12, the blue circles depict T_g.

Now, we need to demonstrate that the set D_t is a *total dominating set* (TDS) of G', satisfying two conditions: (i) every vertex in G' is dominated, and (ii) there is no isolated vertex in $G[D_t]$. Firstly, since w_i and y_i from $T_g \subseteq D_t$ dominate w_i, x_i, y_i, z_i, and p_i, and they are adjacent vertices, $G[T_g]$ does not have any isolated vertex. Secondly, $G[T_g \cup T_{vc}]$ has no isolated vertex since $y_iw_ip_i$ is a path. Now, each p_ip_j correspond to v_iv_j has $4k_{ij}$ vertices and are dominated by $2k_{ij}$ vertices in T_a since out of $4k_{ij}+1$ vertices one vertex is already dominated by a vertex in T_{vc} (as $T_{vc} = \{p_i \in V' : v_i \in S_{vc}\}$). The way the set T_a is con-

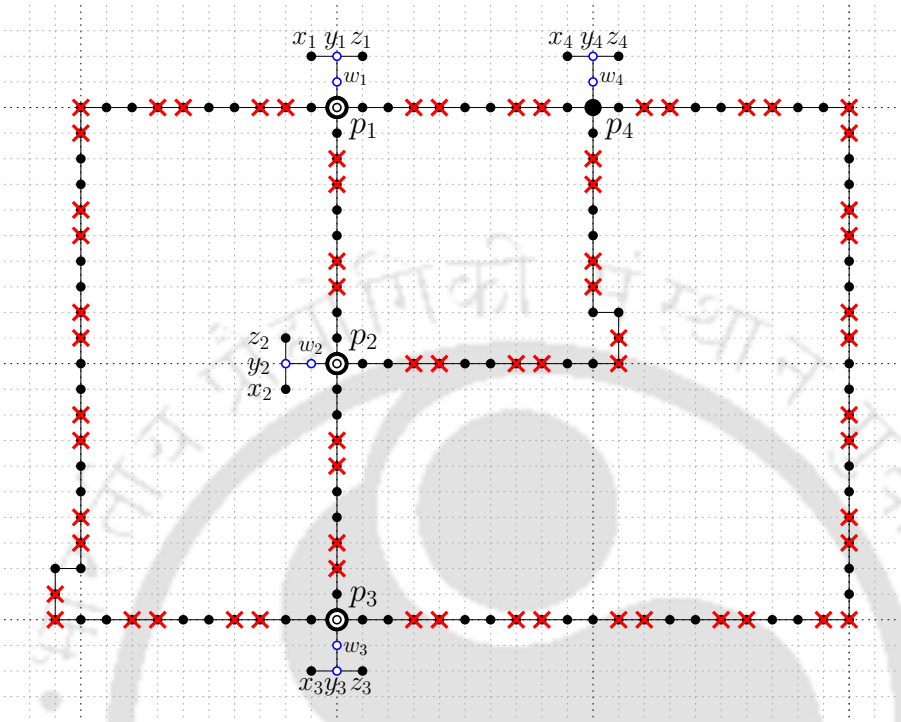

Figure 4.12: $G' = (V', E')$

structed, there does not exist any isolated vertex in $G[T_a]$. Hence, the set $D_t = T_{vc} \cup T_g \cup T_a$ is indeed a TDS of G' and if $S_{vc} \leq k$, then $|D_t| \leq K + 2n + 2\sum_{v_iv_j \in E} k_{ij}$.

($\Longleftarrow$) Let D_t be a TDS of G' such that $|D_t| \leq K + 2n + 2\sum_{v_iv_j \in E} k_{ij}$. Then, we will show that G has a vertex cover S_{vc} of size at most K. To prove this, we prove the following observations.

(i) Out of four points (i.e., x_i, y_i, z_i, w_i) in the gadget associated with each p_i in G', at least two points belong to D_t.

(ii) If p_i and p_j in G' corresponds to the end vertices of an edge $v_iv_j \in E$ and none of p_i and p_j are in D_t, then at least $2k_{ij} + 1$ number of vertices from p_ip_j are in D_t.

Observation (i): Correspond to each $p_i \in E'$, there are 4 points (i.e., x_i, y_i, z_i and w_i) in the corresponding gadget. Since x_i and z_i are pendant vertices, the points y_i and w_i must be in the total dominating set D_t. Hence, in G', $|S \cap D_t| \geq 2n$.

Observation (ii): Since $p_i, p_j \notin D_t$, then there are $4k_{ij}+1$ number of points on p_ip_j for domination. Since two consecutive points can dominate four vertices, $4k_{ij}+1$ number of vertices can be dominated by $\lceil \frac{4k_{ij}+1}{4} \times 2 \rceil = 2k_{ij}+1$.

Given a TDS D_t of G' with size at most $K + 2n + 2\sum_{v_iv_j \in E} k_{ij}$, we need to demonstrate that by deleting and/or replacing some vertices from D_t, we can obtain a *vertex cover* S_{vc} of G with size at most K. The vertices of the set S account for $2n$ vertices in D_t due to Observation (i). Let's define a set $S'_{vc} = D_t \setminus S$ and $S_{vc} = \{v_i \in V : p_i \in S'_{vc}\}$. By Observation (i), we have $|S'_{vc}| \leq K + 2\sum_{v_iv_j \in E} k_{ij}$. Then, for each edge $v_iv_j \in E$, if $p_i, p_j \notin S_{vc}$, then the corresponding p_ip_j in G' has $2k_{ij}+1$ points in D_t instead of $2k_{ij}$ (refer to Observation (ii)). For each such edge, we add either v_i or v_j to S_{vc}. Since every p_ip_j corresponding to each v_iv_j contributes at least $2k_{ij}$ to D_t, $|D_t \cap T_a| \geq 2\sum_{v_iv_j \in E} k_{ij}$. Therefore, $|S_{vc}| \leq K$. This proves that $D\text{-}TDS\text{-}GGs \in NP\text{-}hard$. □

4.5 Conclusion

In this chapter, we have studied the *total dominating set* problem in UDGs and grid graphs. We proposed a 7.79-factor and 6.29-factor approximation algorithms for the TDS problem in UDGs with time complexities $O(n^2)$ and $O(n^2m)$, respectively. In addition, we have also shown that the TDS problem is NP-complete in grid graphs, a subclass of UDGs.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Chapter 5

Total Roman Domination in UDGs

In this chapter, we delve into the *total Roman dominating set* (TRDS) problem in *unit disk graphs* (UDGs). Consider a simple undirected graph $G = (V, E)$ with no isolated vertex, a *total Roman dominating set* is an ordered partition of V, (say (V_0, V_1, V_2), i.e., for $j = 0, 1, 2$ $V_j = \{v \in V : f(v) = j\}$) induced by a function, $f : V \to \{0, 1, 2\}$ such that (i) f is a Roman dominating function (*Roman property*), and (ii) the induced subgraph $G[V_1 \cup V_2]$ has no isolated vertex (*total property*). The weight of the *total Roman dominating function* (TRDF) f is the value $f(V) = \sum_{u \in V(G)} f(u)$. The minimum weight of a TRDF on a graph G is referred to as the *total Roman domination number* and is denoted by $\gamma_{tR}(G)$. The geometric version of the TRDS problem is defined below.

Definition 5. *Given a unit disk graph $G = (V, E)$ with no isolated vertex corresponding to a point set $P = \{p_1, p_2, \ldots, p_n\}$ in $\mathbb{R}^2$, find a total Roman dominating function f of minimum weight.*

A total Roman dominating set of a graph is said to be a *connected Roman dominating set* (CRDS) if the induced sub-graph $G[V_1 \cup V_2]$ forms a single component. The CRDS problem is closely related to the TRDS problem, since every CRDS is inherently a TRDS. In [98], the authors presented a 7.5-factor approximation algorithm for the CRDS problem in UDGs. This algorithm uses a distributed approach detailed in [105], which includes additional message passing overhead. We have observed that the heuristic proposed in [72] can be adapted

to achieve a 12-factor approximation algorithm for the TRDS problem in UDG. Therefore, any constant factor algorithm for the TRDS problem with a factor below 12 and without message passing overhead would be of significant interest.

To address these challenges, we achieve the following goals:

- prove that the TRDS problem is NP-complete in UDGs.
- develop a 10.5-factor approximation algorithm for the TRDS problem in UDGs with time complexity $O(n \log k)$, where n and k are the sizes of the vertex set and the *maximal independent set* of the given UDG, respectively.
- develop a 6.15-factor approximation algorithm for the TRDS problem in UDGs with time complexity $O(n^2m)$, where n and m are the sizes of the vertex set and edge set of the given UDG, respectively.

The remainder of the chapter is organized as follows. In Section 5.1, we list the required preliminaries. In Section 5.2, we prove the hardness result of the TRDS problem in UDGs. In Section 5.3 and Section 5.4, we propose a 10.5-factor and a 6.15-factor approximation algorithms for the UDGs with time complexities $O(n \log k)$ and $O(n^2m)$, respectively. We conclude the chapter in Section 5.5.

5.1 Preliminaries

In this section, we documented several well-known theorems and lemmas. Additionally, we highlighted key observations derived from these theorems and lemmas that are useful for analysis.

Lemma 5.1.1. *[17] If $G = (V, E)$ is a graph without any isolated vertex, then $2\gamma(G) \leq \gamma_{tR}(G) \leq 3\gamma(G)$, where $\gamma(G)$ and $\gamma_{tR}(G)$ are the domination number and total Roman*

domination number of G, respectively.

Observation 5.1.1. *If $f = (V_0, V_1, V_2)$ is a TRDF on G, which attains minimum $|V_1|$, then the induced subgraph on V_1 and V_2 i.e., $G[V_1 \cup V_2]$ does not contain a path[1] of three vertices with Roman value 1 each.*

Proof. Let's assume that there exists a path of three vertices x, y and z in $G[V_1 \cup V_2]$ such that $f(x) = 1$, $f(y) = 1$ and $f(z) = 1$. If this is the case, we can define another TRDF $f' = (V_0', V_1', V_2')$, where $V_0' = V_0 \cup \{x\}$, $V_1' = V_1 \setminus \{x, y\}$ and $V_2' = V_2 \cup \{y\}$ such that $W(f') = W(f)$ and $|V_1'| < |V_1|$. This leads to a contradiction, as it implies that $|V_1|$ is not the minimum. □

For a given graph $G = (V, E)$, let $F^* = \{f_1^*, f_2^*, \ldots, f_{m'}^*\}$ be the set of all TRDFs with $W(f_i^*) = \gamma_{tR}(G)$ for $1 \leqslant i \leqslant m'$. Let $f^* \in F^*$ be a TRDF and let (V_0^*, V_1^*, V_2^*) be the ordered partition of V induced by f^* such that f^* attains minimum $|V_1^*|$. We partition the set V_1^* into subsets V_{11}^* and V_{12}^* such that V_{12}^* is the set of vertices in V_1^* which have neighbors in V_2^* and $V_{11}^* = V_1^* \setminus V_{12}^*$. Similarly, we partition V_2^* into subsets V_{22}^* and V_{21}^* such that V_{21}^* is the set of vertices in V_2^* which have neighbors in V_1^* and $V_{22}^* = V_2^* \setminus V_{21}^*$. In Figure 5.1 (respectively, Figure 5.2), 3 rectangles enclosed within solid line represent the sets V_0^*, V_1^* and V_2^*. The dashed lines in Figure 5.1 (respectively, Figure 5.2) segregate V_1^* into V_{11}^* and V_{12}^*, and V_2^* into V_{22}^* and V_{21}^*.

Lemma 5.1.2. *[3] If $f^* = (V_0^*, V_1^*, V_2^*)$ is an optimal TRDF on a graph $G = (V, E)$ such that f^* attains minimum $|V_1^*|$, then either (i) V_2^* is a dominating set (DS) of G, or (ii) $G[V_{11}^*] = \alpha K_2$ for some integer $\alpha \geq 1$, where K_2 represents the complete graph of two vertices.*

From Lemma 5.1.2, the following observations can be noted.

[1] a sequence of non-repeated vertices connected through edges present in a graph

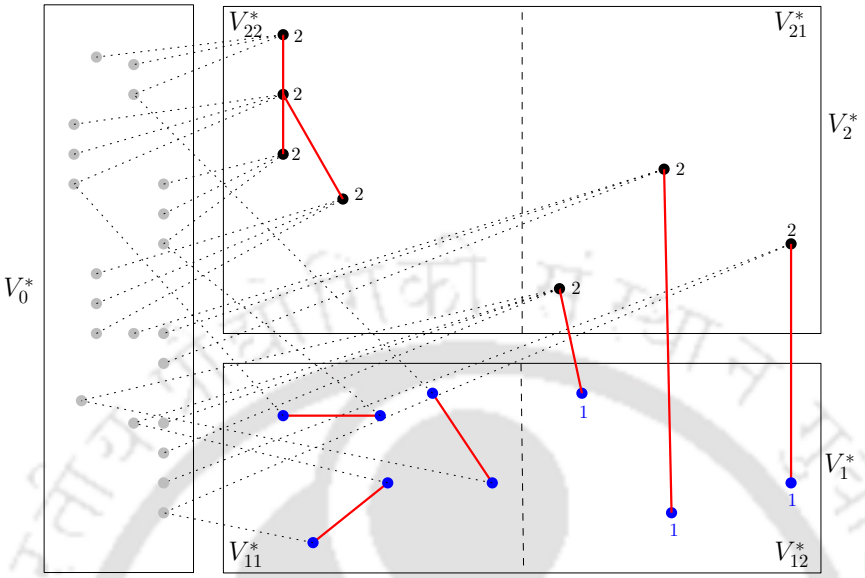

Figure 5.1: *V_2^* is not a DS of G.*

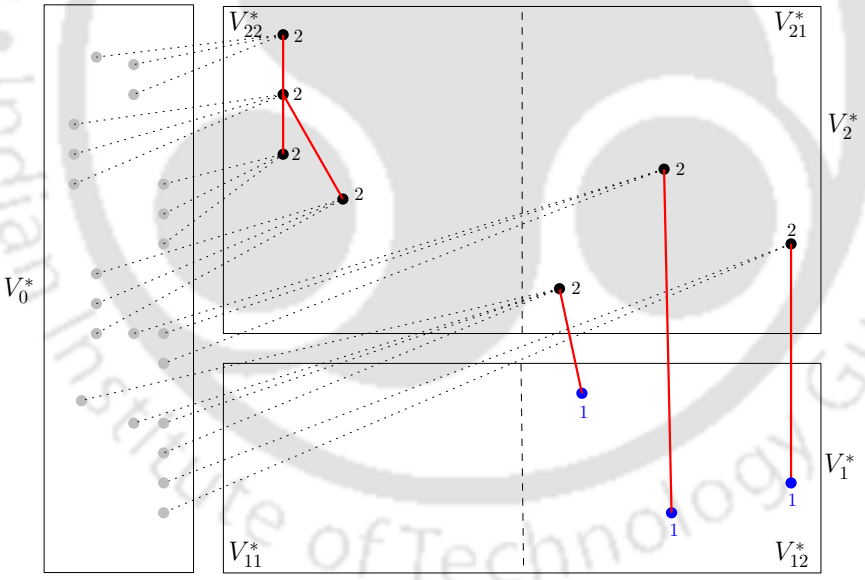

Figure 5.2: *V_2^* is a DS of G.*

Observation 5.1.2. *If $f^* = (V_0^*, V_1^*, V_2^*)$ is an optimal TRDF on the graph $G = (V, E)$ such that f^* attains minimum $|V_1^*|$, then for each edge $pq \in V_{11}^*$, $B(\{p, q\}) \subseteq V_0^*$ (refer to Figure 5.1), where $B(\{p, q\})$ is the boundary of the set $\{p, q\}$*[1].

Proof. It is sufficient to prove that there does not exist any vertex $v \in B(\{p, q\})$ such that

[1]The boundary of a set $S \subseteq V(G)$ is the set $B(S) = N_G(S) \setminus S$

$f(v) = 1$ or $f(v) = 2$. On the contrary, assume that there exists a vertex $v \in B(\{p, q\})$ with $f(v) = 1$. Let $vp \in E(G)$. If so, then we define another optimal TRDF $f^{*\prime} = (V_0^{*\prime}, V_1^{*\prime}, V_2^{*\prime})$, where $V_0^{*\prime} = V_0^* \cup \{v\}$, $V_1^{*\prime} = V_1^* \setminus \{v, p\}$ and $V_2^{*\prime} = V_2^* \cup \{p\}$ such that $W(f^{*\prime}) = W(f^*)$ and $|V_1^{*\prime}| < |V_1^*|$. This leads to a contradiction that $|V_1^*|$ is the minimum.

To prove the second part, assume that there exists a vertex $v \in B(\{p, q\})$ with $f(v) = 2$. Let $vp \in E(G)$. If so, then we define another optimal TRDF $f^{*\prime} = (V_0^{*\prime}, V_1^{*\prime}, V_2^{*\prime})$, where $V_0^{*\prime} = V_0^* \cup \{q\}$, $V_1^{*\prime} = V_1^* \setminus \{p, q\}$ and $V_2^{*\prime} = V_2^* \cup \{p\}$ such that $W(f^{*\prime}) = W(f^*)$ and $|V_1^{*\prime}| < |V_1^*|$. This leads to a contradiction that $|V_1^*|$ is the minimum. □

Observation 5.1.3. *Let $f^* = (V_0^*, V_1^*, V_2^*)$ is an optimal TRDF on a graph $G = (V, E)$ such that f^* attains minimum $|V_1^*|$. If V_2^* is a DS of $G = (V, E)$, then $G[V_1^* \cup V_2^*]$ does not contain an edge with Roman value $(1, 1)$, i.e., $V_{11}^* = \emptyset$ (refer to Figure 5.2).*

Proof. On the contrary, assume $V_{11}^* \neq \emptyset$, i.e., there exists at least one edge $pq \in E(G[V_{11}^*])$ (as f^* satisfies the total property). Since V_2^* is a dominating set of G, each vertex $v \in V(G)$ is either in V_2^* or there exists an edge $uv \in E(G)$ such that $u \in V_2^*$. Since the sets V_2^* and V_{11}^* are mutually exclusive, for each vertex $u \in V_{11}^*$, there exists a vertex $v \in V_2^*$ such that $uv \in E[G]$. Let s and t dominate p and q, respectively, where $s, t \in V_2^*$. If $s = t$ (single vertex which dominates p and q), then we define another TRDF $f^{*\prime} = (V_0^{*\prime}, V_1^{*\prime}, V_2^{*\prime})$, where $V_0^{*\prime} = V_0^* \cup \{q\}$, $V_1^{*\prime} = V_1^* \setminus \{q\}$ and $V_2^{*\prime} = V_2^*$ such that $W(f^{*\prime}) < W(f^*)$. This leads to a contradiction that f^* is optimal; otherwise (i.e., $s \neq t$), p and q would have been in V_{12}^* not in V_{11}^*. □

Observation 5.1.4. *Let $f^* = (V_0^*, V_1^*, V_2^*)$ is an optimal TRDF on a graph $G = (V, E)$ such that f^* attains minimum $|V_1^*|$. If V_2^* is not a DS of G, then $G[V_1^* \cup V_2^*]$ contains edges with Roman values $(2, 2)$, $(2, 1)$ and $(1, 1)$.*

Proof. It follows from Observation 5.1.2 and Observation 5.1.3. □

From Observation 5.1.3 and Observation 5.1.4, we conclude that the induced graph $G[V_1^* \cup$

V_2^*] can have the following *five* structures only: (i) $2-2$: an edge with Roman value 2 for each vertex in the edge, (ii) bouquet of $2-2$: a connected component with Roman value 2 for each vertex in the component, (iii) $2-1$: an edge with Roman value 2 for one vertex and Roman value 1 for another vertex, (iv) flower of $2-1$: a connected component with a star structure, where the center vertex carries Roman value 1, and the remaining carries Roman value 2, (v) $1-1$: an edge with Roman value 1 for each vertex in the edge. Refer to Figure 5.3 for an illustration of each possible type of structure in the induced graph $G[V_1^* \cup V_2^*]$.

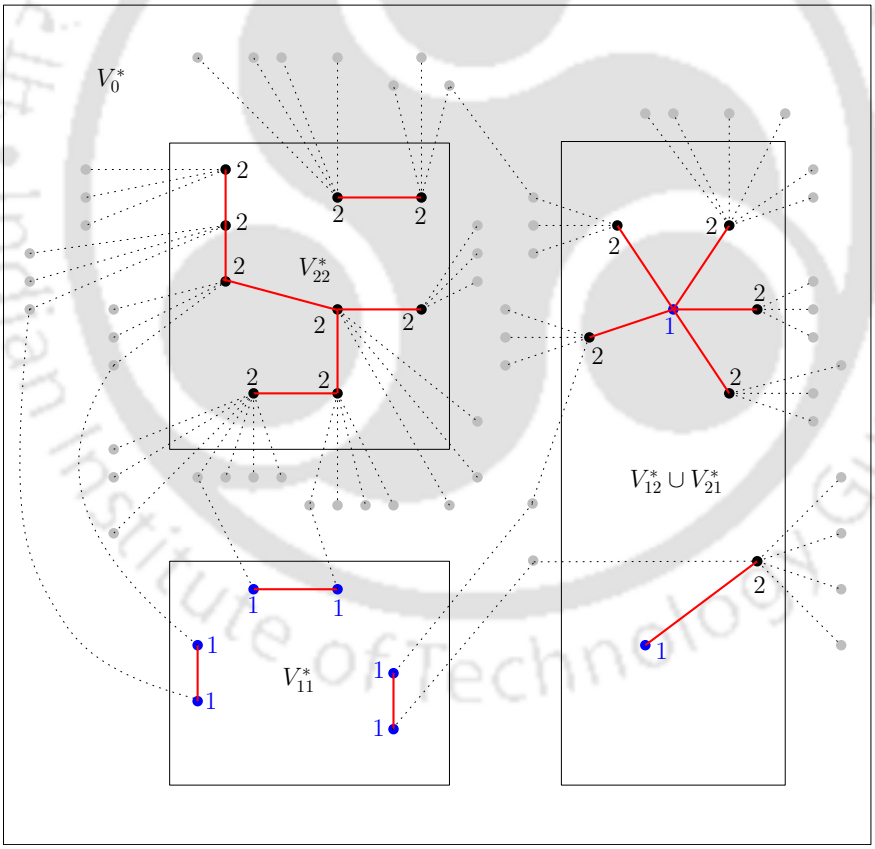

Figure 5.3: *Illustration of* $G[V_{22}^*]$, $G[V_{12}^* \cup V_{21}^*]$, *and* $G[V_{11}^*]$.

5.2 NP-completeness result

In this section, our focus is on determining the complexity of the *total Roman Dominating set* (TRDS) problem. We aim to show that the decision version of the TRDS problem belongs to the NP-complete class when the graph is restricted to UDGs. To accomplish this, we do a reduction from the decision version of the dominating set (DS) problem in grid graphs (GGs) to the decision version of the TRDS problem in unit disk graphs (UDGs). Since the reduction is based on geometry, we often refer to a node or vertex as a 'point'. Formally, the respective decision versions are defined as follows:

The decision version of the DS problem in grid graphs (D-DS-GGs): Given a nonzero positive integer k and a grid graph G, does G have a DS of size at most k?

The decision version of the TRDS problem in UDGs (D-TRDS-UDGs): Given a nonzero positive integer k and a UDG G, does G have a TRDS of weight at most k?

In 1990, Clark et al. [26] showed that D-DS-GGs belongs to the class NP-complete by doing a reduction from the known NP-complete problem planar dominating set of maximum degree 3. We prove the hardness result of D-TRDS-UDGs by making a polynomial time reduction from an arbitrary instance of D-DS-GGs to an instance of D-TRDS-UDGs.

Lemma 5.2.1. *If $G = (V, E)$ is an arbitrary instance of **D-DS-GGs** without any isolated vertex, then an instance $G' = (V', E')$ of **D-TRDS-UDGs** can be constructed from G in polynomial time.*

Proof. Let G be a grid graph with grid size 1×1, where $V = \{p_1, p_2, \ldots, p_n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$ are the vertex set and edge set of G, respectively. We construct a UDG $G' = (V', E')$ from G in two phases as explained below:

In the first phase, for each vertex $p_i \in V$, we add a counterpart vertex $v_i \in V'$, where the distance and orientation between any two vertices $v_i, v_j \in V'$ are exactly the same as their

counterpart vertices $p_i, p_j \in V$. For each edge $p_ip_j \in E$, we address its counterpart v_iv_j as a segment in G'. Let N be the set of nodes added in this phase ($|N| = |V| = n$).

In the second phase, we add some auxiliary nodes in G'. For each segment v_iv_j in G', we add one node x_{ij} on the grid line of the segment v_iv_j at a distance 0.5 either from v_i or from v_j (length of v_iv_j is 1 unit). If p_ip_j is parallel to x-axis and the node x_{ij} is at the coordinate (x, y), then we place another node y_{ij} at coordinate $(x, y+0.1)$ (similarly, if p_ip_j is parallel to y-axis and the node x_{ij} is at the coordinate (x, y), then we place the node y_{ij} at coordinate $(x+0.1, y)$). We refer to each such $x_{ij}y_{ij}$ as a pendant edge. Let A be the set of auxiliary nodes added in this phase to G', i.e., $|A| = 2|E| = 2m$.

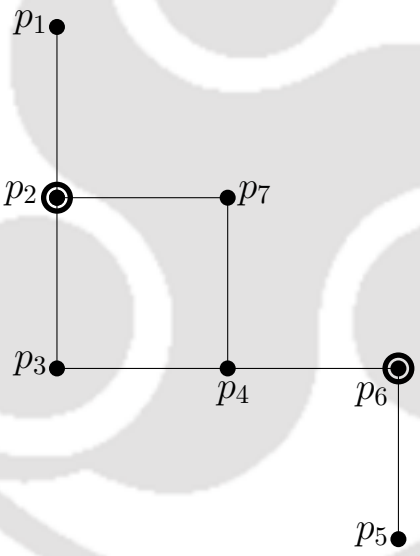

Figure 5.4: $G = (V, E)$

Now, the graph $G' = (V', E')$ with vertex set $V' = N \cup A$ and edge set $E' = \{pq : p, q \in V' \text{ and } \delta(p, q) \leq 0.5\}$ is a *unit disk graph*.

From the preceding steps, it is evident that the UDG G' can be generated from G within polynomial time. Figure. 5.4 and Figure. 5.5(a) provide a comprehensive illustration of the reduction using an example. □

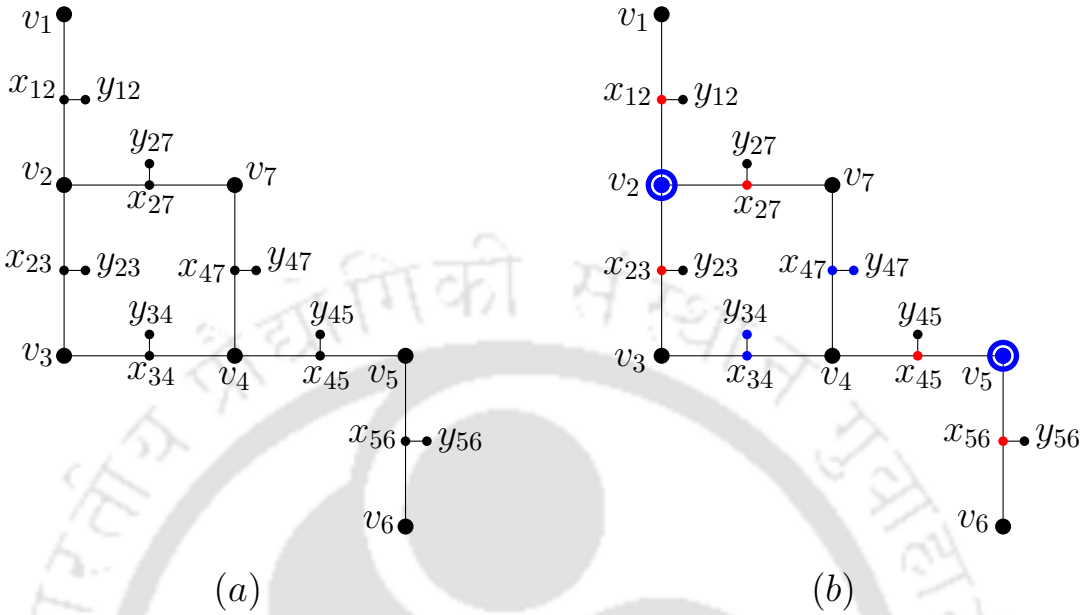

Figure 5.5: *(a)* $G' = (V', E')$, *and (b)* $f = (V_0, V_1, V_2)$

Lemma 5.2.2. *Let* $f = (V_0, V_1, V_2)$ *be a TRDF of* G', *such that* $|V_1|$ *attains minimum value. If* $f(v_i) = 1$, *then for each* j, $f(x_{ij}) = 2$, *where* v_iv_j *is a segment in* G'.

Proof. On the contrary, assume that there exists a segment v_iv_j such that $f(v_i) = 1$ and $f(x_{ij}) = 1$. If so, then $f(y_{ij}) = 1$. We can always have another TRDF f' such that $f'(v_i) = 1$, $f'(x_{ij}) = 2$ and for all other vertices $f(v) = f'(v)$. This leads to a contradiction that $|V_1|$ attains the minimum value in f. □

Theorem 5.2.1. D-TRDS-UDGs *is NP-complete.*

Proof. Given a positive integer k and a graph $G = (V, E)$ with an ordered partition $f = (V_0, V_1, V_2)$ of V, we can verify whether f is a TRDF of G and $W(f) \leq k$ or not in polynomial time. Hence, D-TRDS-UDGs $\in NP$.

To prove D-TRDS-UDGs is NP-hard, we use a polynomial time reduction from D-DS-GGs to D-TRDS-UDGs. First, we construct an instance of D-TRDS-UDGs from an arbitrary instance of D-DS-GGs (i.e., from $G = (V, E)$ to $G' = (V', E')$) using Lemma 5.2.1. Refer to Figure. 5.4, Figure. 5.5(a) and Figure. 5.5(b) for a complete illustration of the reduc-

tion through an example. From Claim 5.2.2.1, $D\text{-}TRDS\text{-}UDGs \in NP\text{-}hard$, therefore, $D\text{-}TRDS\text{-}UDGs \in NP\text{-}complete$. □

Claim 5.2.2.1. *G has a dominating set D of size at most k, if and only if G' has a total Roman dominating function $f = (V_0, V_1, V_2)$ of weight at most $k + 2m$.*

Proof. **Necessity:** To prove this, we consider a TRDF, which attains minimum $|V_1|$; otherwise, for a specific DS of G, there may exist multiple TRDFs for G'.

Let $D \subseteq V$ be a *dominating set* of the grid graph G such that $|D| \leq k$. Now, we assign the Roman value to each vertex $v \in V'$ as listed below:
(i) if $p_i \in D$, then $f(v_i) = 1$, (ii) if $p_ip_j \in E$ and p_i(and/or p_j) $\in D$, then $f(x_{ij}) = 2$, (iii) if $p_ip_j \in E$ and $p_i, p_j \notin D$, then $f(x_{ij}) = f(y_{ij}) = 1$, and (iv) the remaining vertices in G' carry Roman value 0.

Let (V_0, V_1, V_2) be the ordered partition of V' such that $V_i = \{v \in V' : f(v) = i\}$. Now, we argue that $f = (V_0, V_1, V_2)$ is a TRDF. For each edge $p_ip_j \in E$, there exists a counterpart segment v_iv_j in G'. Each segment v_iv_j is associated with an edge $x_{ij}y_{ij}$, where y_{ij} is a pendant vertex as shown in Figure. 5.5(a). So to Roman dominate the vertices of the edge $x_{ij}y_{ij}$, the required Roman value is exactly 2 (either $f(x_{ij}) = 2$ or $f(x_{ij}) = f(y_{ij}) = 1$). If $p_i \in D$, then it dominates each p_j in G, where $p_ip_j \in E$. In contrast, in G', a Roman value 2 to x_{ij} ensures the Roman domination of v_i, v_j and y_{ij}, and a Roman value 1 to v_i ensures the total Roman domination of v_i, v_j and y_{ij}. In each edge $p_ip_j \in E$, if $p_i, p_j \notin D$, the vertices p_i and p_j are dominated by some other vertices, say p_k and p_ℓ, respectively. Therefore, the corresponding counterpart vertices v_i, v_j in G' are Roman dominated by the vertices x_{ik} and $x_{j\ell}$ (i.e., $f(x_{ik}) = 2$ and $f(x_{j\ell}) = 2$) and can be total Roman dominated by assigning Roman value 1 to v_k and v_l (i.e., $f(v_k) = 1$ and $f(v_\ell) = 1$). Hence, the edge $x_{ij}y_{ij}$ associated with the segment v_iv_j needs a Roman value 2 (i.e., $f(x_{ij}) = 1$ and $f(y_{ij}) = 1$) for total Roman domination. Therefore, each segment requires a Roman value of weight 2 for Roman domination. Since there is $|E|$ number of edges in G, which is equal to the number of segments in G', Roman domination of G' requires a weight of at least $2|E|$, and total Roman domination

requires an additional weight with value $|D|$. Hence $W(f) = |D| + 2|E| \leq k + 2m$, where $W(f)$ is the weight associated with the TRDF of G'.

Sufficiency: Let $f = (V_0, V_1, V_2)$ be a TRDF of G' of weight $W(f) \leq k + 2m$. We prove that G has a dominating set D such that $|D| \leq k$.

Given a TRDF $f = (V_0, V_1, V_2)$ of G' with minimum $|V_1|$, we construct a set D as follows: if $v_i \in V_1$, then p_i is in D and if $x_{ij} \in V_2$ and $y_{ij} \in V_1$, then p_j is in D. Now, we show that the set D is a *dominating set* of G. If $f(v_i) = 1$, then for each j, $f(x_{ij}) = 2$, where v_iv_j is a segment in G' (see Lemma 5.2.2). Since $f(x_{ij}) = 2$ ensures the Roman domination of each v_j in G', inclusion of p_i in D ensures the domination of each p_j in G, where $p_ip_j \in E$. Let S_1 be the set containing such p_is. Next, we have to show that for each $v_i \in V'$, if $f(v_i) = 0$, then there exists an edge $p_ip_k \in E$ such that $p_k \in D$. Since f is a TRDF, if $f(v_i) = 0$, then in order to satisfy the *Roman property*, there exists a point $x_{ik} \in V'$ such that $f(x_{ik}) = 2$. In addition to this, it must have satisfied the *total property*, therefore, either $f(y_{ik}) = 1$ or $f(v_k) = 1$. Whatever may be the case, as per the construction of D, $p_k \in D$.

In G', for any segment v_iv_j, if $f(v_i) = f(v_j) = 0$, then $f(x_{ij}) = 2$ and $f(y_{ij}) = 1$, as x_{ij} ensures the Roman domination of v_i, v_j and x_{ij}. However, in G to dominate p_i and p_j, the inclusion of exactly one vertex from p_i and p_j in D is sufficient. So as per the construction of D, p_j ensures the domination of p_i and p_j, where $p_ip_j \in E$. Let S_2 be the set of such p_j's.

From the above two paragraphs, we conclude that $D = S_1 \cup S_2$ is a *dominating set* of G. Now, we are left to show that $|D| \leq k$.

Since each pendant edge corresponding to each segment in G' requires at least a weight of value 2 for Roman domination, the associated weight for total Roman domination in G'

is at least $2|E| = 2m$ (Since the number of segments in G' is exactly equal to the number of edges in G). Since $W(f) \leq k + 2m$, $|D| = |S_1 \cup S_2| \leq k$. Therefore, $D\text{-}TRDS\text{-}UDGs \in NP\text{-}hard$. □

5.3 A 10.5-factor Approximation Algorithm

In this section, we propose a 10.5-factor approximation algorithm called TRDF-UDG for the TRDS problem in geometric UDGs. The algorithm runs on a graph without an isolated vertex. If the graph is disconnected, each component can run it to obtain the TRDF.

5.3.1 Algorithm

Given a UDG $G = (V, E)$, where $V = \{p_1, p_2, \ldots, p_n\} \subseteq \mathbb{R}^2$ is the set of disk centers, the algorithm finds a TRDF $f = (V_0, V_1, V_2)$ of G. First, it finds a *maximal independent set* $V_2 \subseteq V$ of G to satisfy the *Roman property*. Next, to satisfy the *total property*, it chooses a set of neighboring vertices $V_1 \subseteq V \setminus V_2$ such that for each $u \in V_2$, there exists a vertex $u' \in V_1$ and $u' \in U(u)$. See Algorithm 5.1 (*TRDF-UDG*) for the pseudocode, Lemma 5.3.1 for the correctness and Lemma 5.3.3 for time complexity analysis/implementation details of the algorithm.

Lemma 5.3.1. *The function $f = (V_0, V_1, V_2)$ in Algorithm 5.1 is a TRDF of G.*

Proof. Algorithm 5.1 runs in two phases. In the first phase, it finds a maximal independent set V_2 of G (since every maximal independent set is a dominating set) and then assigns Roman value 2 to each vertex in V_2 (see Lines 2-6 of Algorithm 5.1), which ensures the *Roman property* of TRDF. To ensure the *total property* of TRDF, it finds another set V_1 by adding a neighbor vertex for each vertex in V_2 and assigns Roman value 1 (see Lines 7-10 of Algorithm 5.1). The remaining vertices carry Roman value 0 (see Lines 12-14 of Algorithm 5.1). Therefore, combinedly the nominated points in V_2 and V_1 satisfies the *Roman* and *total* properties. Hence, the function $f = (V_0, V_1, V_2)$ is a TRDF of G. □

Lemma 5.3.2. *A cell of size 1×1 contains the centers of at most 3 independent unit disks.*

Algorithm 5.1 TRDF-UDG

Require: A unit disk graph, $G = (V, E)$ with known disk centers
Ensure: A TRDF $f = (V_0, V_1, V_2)$ and the corresponding weight $W(f)$

1: $V_0 = \emptyset$, $V_1 = \emptyset$, $V_2 = \emptyset$, $V' = V$
2: **while** $V' \neq \emptyset$ **do** ▷ Roman property of TRDF
3: choose a vertex $v \in V'$
4: $V_2 = V_2 \cup \{v\}$ and $f(v) = 2$
5: $V' = V' \setminus N_G[v]$
6: **end while**
7: **for** each $u \in V_2$ **do** ▷ total property of TRDF
8: choose a vertex $u' \in N_G(u)$
9: $V_1 = V_1 \cup \{u'\}$ and $f(u') = 1$
10: **end for**
11: $V_0 = V \setminus (V_1 \cup V_2)$
12: **for** each $u \in V_0$ **do**
13: $f(u) = 0$
14: **end for**
15: **return** $f = (V_0, V_1, V_2)$ and $W(f) = 2 \times |V_2| + |V_1|$

Proof. Since the perimeter of a cell is 4 unit, the number of independent unit disks that a cell can contain is at most 3; otherwise, the disks are no longer independent. □

Lemma 5.3.3. *Algorithm 5.1 runs in $O(|V| \log k)$ time, where $k = |V_2|$.*

Proof. Let $V = \{p_1, p_2, \ldots, p_n\}$ be the set of disks' centers corresponding to graph $G = (V, E)$. Let $\mathbb{R}$ be a rectangular plane containing the set of points V, where the extreme left and extreme bottom arms of the rectangle represent the x- and y-axis, respectively. Split the plane $\mathbb{R}$ into horizontal strips and then vertical strips of width one unit each, resulting in a grid of cell size 1×1. Let index each cell as $[x, y]$, where $x, y \in \mathbb{N} \cup \{0\}$. If any point $p \in V$ is located at co-ordinates (p_x, p_y) in the given plane, then it belongs to a cell with index $[\lfloor p_x \rfloor, \lfloor p_y \rfloor]$.

In phase *one*, Algorithm 5.1 constructs a maximal independent dominating set V_2 of the input graph G. To do so efficiently, each non-empty cell maintains a list that keeps the points of V that are chosen for inclusion in V_2, and they are located within that cell. While considering a point $p \in V$ as a candidate for the set V_2, it only probes into 9 cells.

That means if p is located at co-ordinate (p_x, p_y), then it searches in each $[i, j]$ cell, where $\lfloor p_x \rfloor - 1 \leqslant i \leqslant \lfloor p_x \rfloor + 1$ and $\lfloor p_y \rfloor - 1 \leqslant j \leqslant \lfloor p_y \rfloor + 1$.[1] If there does not exist any point $q \in V_2$ in those 9 cells such that $p \in U(q)$, then p is included in V_2. A height balance binary tree containing non-empty cells is used to store the points that are in V_2. Since each cell of size 1×1 can contain at most 3 independent unit disks (see Lemma 5.3.2), the processing time to decide whether a point is in V_2 or not requires $O(\log k)$ time. Thus the time taken to process $|V|$ points is $O(|V| \log k)$, where $k = |V_2|$ (see Lines 2-6 of Algorithm 5.1). In phase *two*, it finds a neighboring vertex u' for each $u \in V_2$ and assigns Roman value 1. Let the set of neighboring vertices be V_1. Since, $|V_1| = |V_2| = k$, the time taken in phase *two* is $O(k)$ (see Lines 7-10 of Algorithm 5.1). Then it assigns Roman value 0 to the remaining vertices of V (excluding V_1 and V_2) in $O(|V|)$ time (see lines 12-14 of Algorithm 5.1). So in total, Algorithm 5.1 runs in $O(|V| \log k)$ time. □

5.3.2 Analysis

Lemma 5.3.4. *Consider two points $p, q \in \mathbb{R}^2$ such that $\delta(p, q) \leq 1$. If there exists a point $r \in \mathbb{R}^2$ such that either $\delta(r, p) \leq 1$ and/or $\delta(r, q) \leq 1$ and S' is the set of independent unit disks that contains at least one point from $\{p, q, r\}$, then $|S'| \leq |S| + 4$, where S is the set of independent unit disks containing p and/or q.*

Proof. Without loss of generality, assume that $\delta(q, r) \leq 1$. Suppose there exists 5 independent unit disks that contain r but neither p nor q, i.e., $|S'| = |S| + 5$. Let c_1, c_2, c_3, c_4 and c_5 be the centers of those 5 unit disks. Let r_i denote the ray $\overrightarrow{rc_i}$ as depicted in Figure 5.6, where $1 \leq i \leq 5$. Without loss of generality, let q lie between the centers c_1 and c_5. Since, $\angle c_1rq + \angle c_5rq > \frac{2\pi}{3}$, so at least one of $\angle c_1rc_2$, $\angle c_2rc_3$, $\angle c_3rc_4$ and $\angle c_4rc_5$ is less than $\frac{\pi}{3}$. This leads to a contradiction of the fact that the disks located at centers c_1, c_2, c_3, c_4, and c_5 are independent. Thus, $|S'| \leq |S| + 4$. □

[1] Any point outside these 9 cells is independent from p

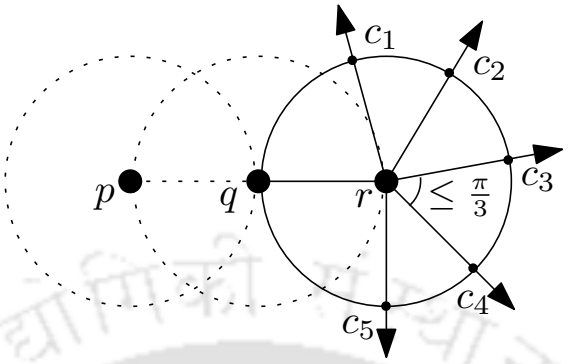

Figure 5.6: *Illustration of independent disks orientation of a pendant vertex.*

Lemma 5.3.5. *Consider two points $p, q \in \mathbb{R}^2$ such that $\delta(p, q) \leq 1$. Let S be the set of independent disks of radius 1 such that S contains the points p and/or q. If $|S| > 7$, then S contains at least one disk having center in $U(p) \cap U(q)$.*

Proof. It is sufficient to prove that if $|S| = 8$, then S contains a disk having its center in $U(p) \cap U(q)$. Let $c_1, c_2, \ldots, c_8$ be the disk centers in S. On the contrary assume that $c_i \notin U(p) \cap U(q)$ for $i = 1, 2, \ldots, 8$. Since $U(c_1), U(c_2), \ldots, U(c_8)$ are independent, without loss of generality, we assume that $c_1, c_2, \ldots, c_8$ lie on the boundary of $U(p) \setminus U(q)) \cup (U(q) \setminus U(p)$. The length of the boundary in $(U(p) \setminus U(q)) \cup (U(q) \setminus U(p))$ is at most $8\pi/3$ as $\delta(p, q) \leq 1$ (refer to Figure 5.7). Since $U(c_1), U(c_2), \ldots, U(c_8)$ are independent, the mutual distance between any two points among $c_1, c_2, \ldots, c_8$ must exceed $\pi/3$, which results in a contradiction. □

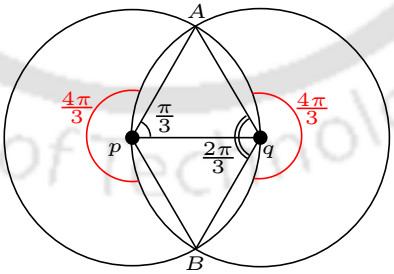

Figure 5.7: *Illustration of boundary length of $U(p) \setminus U(q)) \cup (U(q) \setminus U(p)$, when $\delta(p, q) = 1$.*

Let $f^* = (V_0^*, V_1^*, V_2^*)$ be an optimal TRDF with minimum $|V_1^*|$ and $f = (V_0, V_1, V_2)$ be the solution returned by Algorithm 5.1 (*TRDF-UDG*). From Observation 5.1.3 and Observa-

tion 5.1.4, we know that each edge $pq \in E(G[V_1^* \cup V_2^*])$ has Roman value either $(2,2)$ or $(2,1)$ or $(1,1)$. Let $(V_{11}^*, V_{12}^* \cup V_{21}^*, V_{22}^*)$ be a partition of the set $V_1^* \cup V_2^*$ such that V_{11}^*, V_{12}^*, V_{21}^* and V_{22}^* are the sets defined earlier in Section 5.1. The inner 3 rectangles in Figure 5.3 represent the sets V_{11}^*, $V_{12}^* \cup V_{21}^*$ and V_{22}^*, and the red solid lines in each rectangle illustrate the corresponding induced graph.[1]

Lemma 5.3.6. *If $f^* \in F^*$ is an optimal TRDF, which attains minimum $|V_1^*|$, then Algorithm 5.1 charges at most weight 24 against an edge $pq \in E(G[V_{22}^*])$.*

Proof. Let $f = (V_0, V_1, V_2)$ be the TRDF returned by Algorithm 5.1 with weight $W(f)$. Consider an arbitrary edge $pq \in E(G[V_{22}^*])$. From Lemma 4.1.1, there are at most 8 independent unit disks in V_2 which contain the points p and/or q. To satisfy the Roman property of TRDF, Algorithm 5.1 assigns Roman value 2 (see line 4 in Algorithm 5.1) to each of the 8 vertices, which are the elements of V_2. So, our algorithm may invest $2 \times 8 = 16$ from $W(f)$ to dominate (Roman) both p and q. Next, to satisfy the total property, Algorithm 5.1 chooses a neighbor for each selected vertex in phase *one* from V_1 and assigns Roman value 1 (see line 9 in Algorithm 5.1), which requires an additional 8 investment from $W(f)$. So, in total, Algorithm 5.1 invests at most 24 from $W(f)$ weight. So for any edge $pq \in E(G[V_{22}^*])$ having Roman value $(2,2)$, $f = (V_0, V_1, V_2)$ dominates (total Roman) p and q by investing at most 24 weight from $W(f)$. □

Lemma 5.3.7. *If $f^* \in F^*$ is an optimal TRDF which attains minimum $|V_1^*|$, then TRDF-UDG charges at most weight 24 against an edge $(p,q) \in E(G[V_{12}^* \cup V_{21}^*])$.*

Proof. Proof of the lemma is similar to Lemma 5.3.6. □

Lemma 5.3.8. *Let $f^* = (V_0^*, V_1^*, V_2^*)$ be an optimal TRDF, which attains minimum $|V_1^*|$. Let S be a set of independent unit disks such that each disk in S contains p and/or q, where $pq \in E(G[V_{11}^*])$. Then, $|S| \leq 7$.*

[1] In Figure 5.3, the grey-colored (lightly shaded) disks represent the vertices in V_0^* and the dotted lines represent the edges between the set V_0^* and $V_1^* \cup V_2^*$

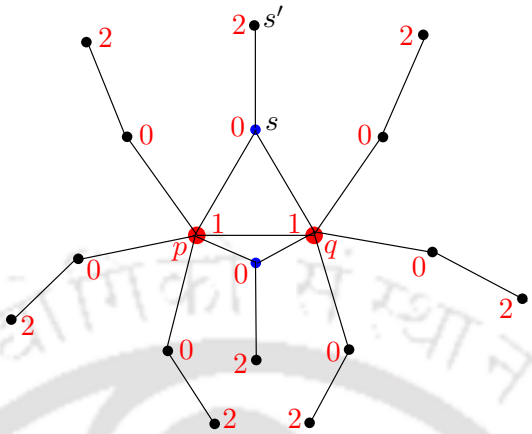

Figure 5.8: *Illustration of Lemma 5.3.8.*

Proof. Let $|S| \geq 8$. So, by Lemma 5.3.5, there exists at least one disk in S having its center in $U(p) \cap U(q)$. Let the disk's center be s (see Figure 5.8). From Observation 5.1.2, $f^*(s) = 0$. This implies there exist a vertex $s' \in N_G(s)$ such that $f^*(s') = 2$. Now, we define another optimal TRDF $f^{*\prime} = (V_0^{*\prime}, V_1^{*\prime}, V_2^{*\prime})$, where $V_0^{*\prime} = V_0^* \setminus \{s\} \cup \{p, q\}$, $V_1^{*\prime} = V_1^* \setminus \{p, q\}$ and $V_2^{*\prime} = V_2^* \cup \{s\}$ such that $W(f^{*\prime}) = W(f^*)$ and $|V_1^{*\prime}| < |V_1^*|$. This leads to a contradiction that $|V_1^*|$ is the minimum. □

Lemma 5.3.9. *If $f^* \in F^*$ is an optimal TRDF which attains minimum $|V_1^*|$, then Algorithm 5.1 charges at most* 21 *against an edge $pq \in E(G[V_{11}^*])$.*

Proof. Let $f = (V_0, V_1, V_2)$ be the TRDF returned by Algorithm 5.1 with weight $W(f)$. Consider an arbitrary edge $pq \in E(G[V_{11}^*])$. From Lemma 5.3.8, there are at most 7 independent disks that contain p and/or q. So, in the worst case, at most 7 independent disks in V_2 contain the point p and/or q. The Roman value assigned to each vertex in V_2 is 2 (see Line 4 in Algorithm 5.1). So, Algorithm 5.1 may invest $2 \times 7 = 14$ from $W(f)$ to dominate (Roman) both p and q. Now, to satisfy *total property*, Algorithm 5.1 further selects a neighbor for each $v \in V_2$ from V_1 and assigns the Roman value 1 (see Line 9 in Algorithm 5.1), which requires additional 7 investment from $W(f)$. So, in total, Algorithm 5.1 invests at most 21 from $W(f)$ weight. So for any edge $pq \in E(G[V_{11}^*])$ having Roman value $(1, 1)$, $f = (V_0, V_1, V_2)$ dominates (total Roman) p and q by investing at most 21 weight from $W(f)$. □

Lemma 5.3.10. *Let A ($|A| = l$) be the set of vertices in a bouquet/flower (mentioned in Section 5.1) and S be a set of independent unit disks that contain the vertices in A. Then, $|S| \leq 4l$.*

Proof. Consider an edge pq in the bouquet/flower. Then there exists at most 8 independent unit disks that can contain p and/or q (by Lemma 4.1.1). Now consider another vertex r attached either to p or q, then at most, 4 independent unit disks can contain only r (by Lemma 5.3.4). So, at most, $8 + 4 = 12$ independent disks can contain p and/or q and/or r. Let $A' \subseteq A$ be a set of vertices in the bouquet/flower apart from p and q, i.e., $A' = A \setminus \{p, q\}$. Note that $|A'| = l - 2$. So, by Lemma 5.3.4, there exists at most 4 independent disks for each vertex in A'; thus, at most $8 + 4(l - 2) = 4l$ independent disks can contain all the vertices in A. □

Let C^* be a connected component in $G[V_1^* \cup V_2^*]$. For any component C^*, we denote $W(C^*)$ and $W^*(C^*)$ as the weights incurred by Algorithm 5.1 and f^*, respectively.

Lemma 5.3.11. *If $f^* = (V_0^*, V_1^*, V_2^*)$ is an optimal TRDF such that $|V_1^*|$ attains minimum value and $f = (V_0, V_1, V_2)$ is the TRDF returned by Algorithm 5.1, then for each component $C^* \in G[V_1^* \cup V_2^*]$, $W(C^*) \leq 10.5 \times W^*(C^*)$.*

Proof. Let $f^* = (V_0^*, V_1^*, V_2^*)$ be an optimal TRDF which attains minimum $|V_1^*|$. Consider the ordered partition $(V_{11}^*, V_{12}^* \cup V_{21}^*, V_{22}^*)$ of the set $V_1^* \cup V_2^*$ as defined in Section 5.1. From Section 5.1, the induced graph $G[V_1^* \cup V_2^*]$ exhibits *five* different structures only (refer to Figure 5.3). To make the analysis easier, we consider each of the induced graphs $G[V_{22}^*]$, $G[V_{12}^* \cup V_{21}^*]$ and $G[V_{11}^*]$ separately.

Case 1: $G[V_{22}^*]$

By Lemma 5.3.10, there are $4l$ independent disks that can contain vertices of a bouquet. So, in the first phase, Algorithm 5.1 can pick at most $4l$ independent disks. Algorithm 5.1 invests $3 \times 4l$ weight from $W(f)$. But in the optimal, $2l$ weight out of $W(f^*)$ is required for the bouquet (since each vertex carries Roman value 2). So, Algorithm 5.1 invests $\frac{12l}{2l} = 6$

times more if compared to the optimal solution f^*.

Case 2: $G[V_{12}^* \cup V_{21}^*]$
By Lemma 5.3.10, There are $4l$ independent disks that can contain vertices of a flower. So, in the first phase, Algorithm 5.1 can pick at most $4l$ independent disks. *TRDF-UDG* invests $3 \times 4l$ weight from $W(f)$. But in the optimal, $3+2(l-2) = 2l-1$ weight out of $W(f^*)$ is required for the flower. Note that $\frac{12l}{2l-1}$ is maximum when the flower consists of only $l = 2$ vertices, and in f^*, the Roman value assigned to the two vertices are 2 and 1. The more is the number of vertices in a flower; the less is the value of $\frac{12l}{2l-1}$ (since the function $\frac{12l}{2l-1}$ is a monotonically decreasing function). So, at maximum, Algorithm 5.1invests 8 times more if compared to the optimal solution f^*.

Case 3: $G[V_{11}^*]$
From Lemma 5.1.2, each component $C^* \in G[V_{11}^*]$ is a K_2. Algorithm 5.1 invests at most 21 against C^* by Lemma 5.3.9. Hence, Algorithm 5.1 invests $\frac{21}{2} = 10.5$ times more for $C^* \in G[V_{11}^*]$ in f compared to f^*. □

Lemma 5.3.12. *$W(f) \leq 10.5 \times \gamma_{tR}(G)$, where $W(f)$ and $\gamma_{tR}(G)$ are the weights associated with f (in Algorithm 5.1) and f^* (optimal TRDF) of G, respectively.*

Proof. Since $V_{11}^* \cup (V_{12}^* \cup V_{21}^*) \cup V_{22}^* = V_1^* \cup V_2^*$, therefore, the theorem follows from Lemma 5.3.11. □

Theorem 5.3.1. *The proposed algorithm (Algorithm 5.1) gives a 10.5-factor approximation algorithm for the TRDS problem in UDGs, which runs in $O(|V| \log k)$ time, where k is the number of vertices with Roman value 2.*

Proof. The approximation factor follows from Lemma 5.3.12, and the result of time complexity follows from Lemma 5.3.3. □

5.4 A 6.15-factor Approximation Algorithm

In this section, we propose a 6.15 - factor approximation algorithm for the TRDS problem in geometric UDGs.

5.4.1 Algorithm

Now, we describe the procedure for finding a TRDF of a UDG $G = (V, E)$. First, we find a maximal independent set $V_2 \subseteq V$ of G to satisfy the Roman property. Next, to satisfy the total property, we choose a set of neighboring vertices $V_1 \subseteq V$ such that for each $v \in V_2$, there exists at least one vertex $u \in V \setminus V_2$ and $u \in U(v)$. The steps of finding the sets V_2 and V_1 in this algorithm are similar to those of finding the sets D and T in Algorithm 4.2, respectively. See Algorithm 5.2 (TRDF-UDG-SC) for the complete pseudocode, Lemma 5.4.1 for the correctness and Lemma 5.4.2 for the time complexity of the algorithm.

Algorithm 5.2 TRDF-UDG-SC

Input: A unit disk graph, $G = (V, E)$ with known disk centers
Output: A TRDF $f = (V_0, V_1, V_2)$ and the corresponding weight $W(f)$

$V_0 = \emptyset$, $V_1 = \emptyset$, $V_2 = \emptyset$, $V' = V$
while $V' \neq \emptyset$ **do** ▷ *Roman* property of TRDF
    choose a vertex $v \in V'$
    $V_2 = V_2 \cup \{v\}$
    $V' = V' \setminus N_G[v]$
end while
$i = 1$, $S = \emptyset$
for each $u_i \in V \setminus V_2$ **do**
    $S_i = N_G(u_i) \cap V_2$
    $i = i + 1$, $S = \emptyset$
    $S = S \cup S_i$
end for
$S' = GreedySetCover(V_2, S)$
$V_1 = \{u_i | S_i \in S'\}$ ▷ *total* property of TRDF
$V_0 = V \setminus (V_1 \cup V_2)$
return $f = (V_0, V_1, V_2)$ and $W(f) = 2 \times |V_2| + |V_1|$

Lemma 5.4.1. *The function $f = (V_0, V_1, V_2)$ in Algorithm 5.2 is a TRDF on G.*

Proof. Algorithm 5.2 runs in two phases. In the first phase, it finds a maximal independent set V_2 of G (since every maximal independent set is a dominating set) and then assigns Roman value 2 to each vertex in V_2 (see Lines 2-6 of Algorithm 5.2), which ensures the *Roman property* of TRDF. To ensure the *total property* of TRDF, the algorithm uses a function $GreedySetCover(V_2, S)$ to identify another set V_1 in $V \setminus V_2$ by adding a vertex $u_i \in V \setminus V_2$ corresponding to each $S_i \in S'$ (see Line 13 and Line 14 of Algorithm 5.2). The remaining vertices in G carry Roman value 0 (see Line 15 of Algorithm 5.2). Thus, the combination of the vertices in V_2 and V_1 ensures both the *Roman* and *total* properties. Consequently, the function $f = (V_0, V_1, V_2)$ forms a TRDF on G. □

Lemma 5.4.2. *Algorithm 5.2 runs in $O(n^2m)$ time, where n and m are the number of vertices and edges in G.*

Proof. The proof is similar to Lemma 5.4.2. □

5.4.2 Analysis

Let D^* be an optimal *dominating set* and f^* be an optimal TRDF of G, where $W(f^*)$ is the weight associated with f^*. Let C^* be an optimal *set cover* of the set cover instance $< V_2, S >$. Since V_2 is a *maximal independent set*, from Lemma 3.1.2, we have the following equation.

$$|V_2| \leq 5|D^*| \tag{5.1}$$

From Lemma 5.1.1, we conclude that

$$2|D^*| \leq W(f^*) \tag{5.2}$$

Since $f = (V_0, V_1, V_2)$ is a TRDF on G (see Lemma 5.4.1), the associated weight $W(f)$ of the TRDF f is given by:

$$
\begin{aligned}
W(f) &= 2 \times |V_2| + |V_1| \\
&\leq 2 \times 5|D^*| + H(5) \times |C^*| \text{ (refer to Equation 5.1 and Theorem 4.1.1)} \\
&\leq 5 \times 2|D^*| + \frac{137}{60} \times |D^*| \text{ (refer to Equation 4.5 and Lemma 4.3.1)} \\
&\leq 5 \times 2|D^*| + \frac{137}{120} \times 2|D^*| \qquad (5.3) \\
&\leq 5 \times W(f^*) + \frac{137}{120} \times W(f^*) \text{ (refer to Equation 5.2)} \\
&= \frac{737}{120} \times W(f^*) \\
&\leq 6.15 \times W(f^*)
\end{aligned}
$$

Theorem 5.4.1. *The proposed algorithm (TRDF-UDG-SC) gives a* 6.15*- factor approximation result for the TRDF problem in UDGs. The algorithm runs in* $O(n^2m)$ *time, where* n *and* m *are the number of vertices and edges of the graph.*

Proof. The approximation factor and the time complexity result follow from Equation 5.3 and Lemma 5.4.2, respectively. □

5.5 Conclusion

In this paper, we studied the *total Roman dominating set* problem in UDGs and proved that the problem is NP-complete in UDGs. We proposed a 10.5- factor and a 6.15- factor approximation algorithm for the TRDS problem in UDGs with time complexities $O(n \log k)$ and $O(n^2m)$, respectively, where n is the number of vertices, m is the number of edges in G and k is the size of the set V_2.

Chapter 6

Total Roman Domination in Trees

In this chapter, we study the *total Roman dominating set* problem in trees.

Definition 6. *Given a tree $T = (V, E)$, find a total Roman dominating function (TRDF) f of minimum weight.*

The objective of this chapter is to design a linear-time algorithm to find the minimum total Roman dominating set in trees.

The remainder of this chapter is organized as follows. In Section 6.1, we list the required preliminaries. In Section 6.2, we propose a linear-time algorithm to compute the TRDF of a tree. Finally, we conclude the chapter in Section 6.3.

6.1 Preliminaries

Let $T = (V, E)$ represent a tree rooted at a vertex $r \in V$, where V and E are the sets of vertices and edges of the tree T, respectively. In a tree, a parent of a vertex $v \in V$ is the vertex $u \in V$ connected to v by an edge on the path to the root. Every vertex has a unique parent except for the root, which has no parent. A child of a vertex u is a vertex for which u is the parent. Let $C(u)$ denote the set of children of a vertex $u \in V$.

6.2 A Linear Time Algorithm

In this section, we use dynamic programming to find the *total Roman domination number* of a tree $T = (V, E)$. Let u be a specific vertex in T, and let $T' = (V', E')$ be the subtree rooted at u. Let $f : V' \to \{0, 1, 2\}$ be a TRDF for the subtree T'. For the vertex u, it holds that $f(u) \in \{0, 1, 2\}$. Thus, it is useful to consider the following six subproblems:

(i) $dp_0[u] = \min\{\ f(V')\ :\ f(u) = 0\ \ and\ \ f(v) \in \{0, 1, 2\}\ for\ all\ \ v \in C(u)\}$

(ii) $dp_0^2[u] = \min\{\ f(V')\ :\ f(u) = 0,\ \ f(v) = 2\ for\ some\ v \in C(u),$
$and\ f(x) \in \{0, 1, 2\}\ , for\ all\ x \in C(u) \setminus \{v\}\}$

(iii) $dp_1[u] = \min\{\ f(V')\ :\ f(u) = 1\ \ and\ \ f(v) \in \{0, 1, 2\}\ for\ all\ \ v \in C(u)\}$

(iv) $dp_1^{1,2}[u] = \min\{\ f(V')\ :\ f(u) = 1,\ \ f(v) \in \{1, 2\}\ for\ some\ v \in C(u),$
$and\ f(x) \in \{0, 1, 2\}\ , for\ all\ x \in C(u) \setminus \{v\}\}$

(v) $dp_2[u] = \min\{\ f(V')\ :\ f(u) = 2\ \ and\ \ f(v) \in \{0, 1, 2\}\ for\ all\ \ v \in C(u)\}$

(vi) $dp_2^{1,2}[u] = \min\{\ f(V')\ :\ f(u) = 2,\ \ f(v) \in \{1, 2\}\ for\ some\ v \in C(u),$
$and\ f(x) \in \{0, 1, 2\}\ for\ all\ x \in C(u) \setminus \{v\}\}$

In this context, $dp_0[u]$ represents the total Roman domination number (TRDN) for the subtree T' when $f(u) = 0$, where all children of u can take any Roman value from $0, 1, 2$ under the TRDF. Similarly, $dp_1[u]$ and $dp_2[u]$ denote the TRDN for the subtree T' when $f(u) = 1$ and $f(u) = 2$, respectively, under TRDF. Moreover, $dp_0^2[u]$ stands for the TRDN of the subtree T' when $f(u) = 0$, with one child of u assigned the Roman value 2, while all other children of u can take any Roman value from $\{0, 1, 2\}$ under TRDF. Note that in $dp_0^2[u]$, the children meet the first TRDF condition (Roman property). Furthermore, $dp_1^{1,2}[u]$ represents the TRDN for T' when $f(u) = 1$, with one child of u assigned either the Roman value 1 or 2, and all other children of u allowed any Roman value from $\{0, 1, 2\}$ under TRDF. Likewise, $dp_2^{1,2}[u]$ stands for the TRDN of T' when $f(u) = 2$, with one child of u assigned either the Roman value 1 or 2, and the remaining children of u having any

Roman value from $\{0, 1, 2\}$ under TRDF. Note that in both $dp_1^{1,2}[u]$ and $dp_2^{1,2}[u]$, the second condition of TRDF (total property) is satisfied by the children.

6.2.1 Algorithm

Theorem 6.2.1. *Let $T = (V, E)$ be a tree with a root $r \in V$. Let $u \in V$ be a non-leaf vertex of the tree. Then the following statements hold:*

(i) $dp_0[u] = \sum_{c \in C(u)} \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\}$

(ii) $dp_0^2[u] = \min_{c \in C(u)} \{dp_2^{1,2}[c] + \sum_{c' \in C(u) \setminus \{c\}} \min\{dp_0^2[c'], dp_1^{1,2}[c'], dp_2^{1,2}[c']\}\}$

(iii) $dp_1[u] = 1 + \sum_{c \in C(u)} \min\{dp_0^2[c], dp_1[c], dp_2[c]\}$

(iv) $dp_1^{1,2}[u] = 1 + \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} + \sum_{c' \in C(u) \setminus \{c\}} \min\{dp_0^2[c'], dp_1[c'], dp_2[c']\}\}$

(v) $dp_2[u] = 2 + \sum_{c \in C(u)} \min\{dp_0[c], dp_1[c], dp_2[c]\}$

(vi) $dp_2^{1,2}[u] = 2 + \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} + \sum_{c' \in C(u) \setminus \{c\}} \min\{dp_0[c'], dp_1[c'], dp_2[c']\}\}$

Proof. The proofs of the above statements are as follows:

(i) In this case, since the vertex u has been designated the Roman value 0 by the TRDF, for any child $c \in C(u)$ if $f(c) = 0$, it is imperative to fulfill the first condition of total Roman domination. This necessitates that the TRDF assign at least one child of c the Roman value 2. Conversely, if $f(c) = 1$ or $f(c) = 2$, adhering to the second condition of total Roman domination requires that at least one child of c be assigned the Roman value 1 or 2 by the TRDF. To address the constraint of dp_0, we need to evaluate the minimum for each child $c \in C(u)$, considering the cases where $f(c) = 0$, $f(c) = 1$, and $f(c) = 2$.

(ii) Similar to the previous case, in this case too, as $f(u) = 0$, if there is a child $c \in C(u)$, such that $f(c) = 0$, then to satisfy the first condition of Roman domination, one of the

children of c must be assigned 2 by the RDF. If $f(c) = 1$ or $f(c) = 2$, the TRDF must assign one of the children of c with 1 or 2 to satisfy the second condition of total Roman domination. Also, from the constraint of dp_0^2, at least one of the children say $c \in C(u)$ of u must be assigned 2 by the TRDF to satisfy the total Roman domination first condition, one of the children of c must be assigned 1 or 2 by TRDF to satisfy total Roman domination second condition. Hence, to calculate the total Roman domination number of the subtree rooted at u, we take the minimum of all the cases in which one child $c \in C(u)$ is assigned $f(c) = 2$, and the rest of them are assigned any value from $\{0, 1, 2\}$ by the TRDF.

(iii) In this scenario, vertex u is designated the Roman value 1 by the TRDF. Consequently, for any child $c \in C(u)$, if $f(c) = 0$, it is necessary to fulfill the first condition of total Roman domination by assigning one of c's children the value 2 via the TRDF. Conversely, if $f(c) = 1$ or $f(c) = 2$, since the parent satisfies the second condition of total Roman domination, c's children can be assigned any value by the TRDF. Considering the constraint of dp_1, we evaluate the minimum for each child $c \in C(u)$ under the cases $f(c) = 0$, $f(c) = 1$, and $f(c) = 2$.

(iv) Similar to the previous case, in this case too, as $f(u) = 1$, if there is a child $c \in C(u)$, such that $f(c) = 0$, then to satisfy the first condition of Roman domination, one of the children of c must be assigned 2 by the RDF. If $f(c) = 1$ or $f(c) = 2$, as the second condition of total Roman domination is satisfied by its parent, children of c can be assigned any value by the TRDF. Also, from the constraint of $dp_1^{1,2}$, at least one of the children of u must be assigned 1 or 2 by the TRDF to satisfy the second condition of the total Roman domination. Hence, to calculate the total Roman domination number of the subtree rooted at u, we take the minimum of all the cases in which one child $c \in C(u)$ is assigned either $f(c) = 1$ or $f(c) = 2$, and the rest of them are assigned any value from $\{0, 1, 2\}$ by the TRDF.

(v) In this case, as the vertex u is assigned the value 2 by the TRDF, hence, for any child $c \in C(u)$, if $f(c) = 0$, as the first condition of total Roman domination is satisfied

by its parent, children of c can be assigned any value by the TRDF else if $f(c) = 1$ or $f(c) = 2$, as the second condition of the total Roman domination is satisfied by its parent, children of c can be assigned any value by the TRDF. From the constraint of dp_2, for each child, $c \in C(u)$, we consider the minimum for the cases $f(c) = 0$, $f(c) = 1$ and $f(c) = 2$.

(vi) Similar to the previous case, in this case too, as $f(u) = 2$, if there is a child $c \in C(u)$, such that $f(c) = 0$, as the first condition of total Roman domination is satisfied by its parent, children of c can be assigned any value by the TRDF. If $f(c) = 1$ or $f(c) = 2$, as the second condition of the total Roman domination is satisfied by its parent, children of c can be assigned any value by the TRDF. Also, from the constraint of $dp_2^{1,2}$, at least one of the children of u must be assigned 1 or 2 by the TRDF to satisfy the total Roman domination second condition. Hence, to calculate the total Roman domination number of the subtree rooted at u, we take the minimum of all the cases in which one child $c \in C(u)$ is assigned $f(c) = 1$ or $f(c) = 2$, and the rest of them are assigned any value from $\{0, 1, 2\}$ by the TRDF. □

The outline of the algorithm is as follows. We first do a depth first search (DFS) traversal on the given tree $T = (V, E)$, starting from the root $r \in V$. In the DFS traversal, for a particular vertex, we first visit all the children of the vertex, and then come back to the vertex. Hence, when we reach a vertex v, then the entries corresponding to the children of v are already filled in the 6 arrays, $dp_0, dp_1, dp_2, dp_0^2, dp_1^{1,2}$ and $dp_2^{1,2}$, and hence, from these, we can fill in the values of $dp_0[v]$, $dp_1[v]$, $dp_2[v]$, $dp_0^2[v]$, $dp_2^{1,2}[v]$, and $dp_2^{1,2}[v]$. After the traversal is complete, all the entries of the 6 arrays will be filled. Finally, as the root r can be assigned the values $\{0, 1, 2\}$ by the TRDF, and if $f(r) = 0$, then at least one of the children of r should be assigned 2 to satisfy the TRDF first condition. If $f(r) = 1$ or $f(r) = 2$, then at least one of the children of r should be assigned 1 or 2 to satisfy the TRDF second condition. Hence we take the $\min\{dp_0^2[r], dp_1^{1,2}[r], dp_2^{1,2}[r]\}$ to find out the total Roman domination number for the tree T.

We make some changes to the formulae of $dp_0^2[u]$, $dp_1^{1,2}[u]$ and $dp_2^{1,2}[u]$ for computational efficiency.

Lemma 6.2.1. *For a non-leaf vertex $u \in V$, we can write*

$$dp_0^2[u] = \alpha + dp_0[u]$$

where $\alpha = \min_{c \in C(u)} \{dp_2^{1,2}[c] - \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\}\}$

Proof. From Theorem 6.2.1, we can write,

$$\begin{aligned}
dp_0^2[u] &= \min_{c \in C(u)} \{dp_2^{1,2}[c] + \sum_{c' \in C(u) \setminus c} \min\{dp_0^2[c'], dp_1^{1,2}[c'], dp_2^{1,2}[c']\}\} \\
&= \min_{c \in C(u)} \{dp_2^{1,2}[c] - \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\} + \sum_{c' \in C(u)} \min\{dp_0^2[c'], dp_1^{1,2}[c'], dp_2^{1,2}[c']\}\} \\
&= \min_{c \in C(u)} \{dp_2^{1,2}[c] - \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\} + dp_0[u]\} \\
&= \min_{c \in C(u)} \{dp_2^{1,2}[c] - \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\}\} + dp_0[u] \\
&= \alpha + dp_0[u] \\
&\text{where } \alpha = \min_{c \in C(u)} \{dp_2^{1,2}[c] - \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\}\}
\end{aligned}$$

□

Lemma 6.2.2. *For a non-leaf vertex $u \in V$, we can write*

$$dp_1^{1,2}[u] = \beta + dp_1[u]$$

where $\beta = \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0^2[c], dp_1[c], dp_2[c]\}\}$

Proof. From Theorem 6.2.1, we can write,

$$\begin{aligned}
dp_1^{1,2}[u] &= 1 + \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} + \sum_{c' \in C(u) \setminus \{c\}} \min\{dp_0^2[c'], dp_1[c'], dp_2[c']\}\} \\
&= 1 + \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - \min\{dp_0^2[c], dp_1[c], dp_2[c]\}
\end{aligned}$$

$$+ \sum_{c' \in C(u)} \min\{dp_0^2[c'], dp_1[c'], dp_2[c']\}\}$$

$$= \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0^2[c], dp_1[c], dp_2[c]\} + dp_1[u]\}$$

$$= \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0^2[c], dp_1[c], dp_2[c]\}\} + dp_1[u]$$

$$= \beta + dp_1[u]$$

$$where\ \beta = \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0^2[c], dp_1[c], dp_2[c]\}\}$$

□

Lemma 6.2.3. *For a non-leaf vertex $u \in V$, we can write*

$$dp_2^{1,2}[u] = \gamma + dp_2[u]$$

where $\gamma = \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0[c], dp_1[c], dp_2[c]\}\}$

Proof. From Theorem 6.2.1, we can write,

$$dp_1^{1,2}[u] = 2 + \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} + \sum_{c' \in C(u) \setminus \{c\}} \min\{dp_0[c'], dp_1[c'], dp_2[c']\}\}$$

$$= 2 + \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - \min\{dp_0[c], dp_1[c], dp_2[c]\}$$

$$+ \sum_{c' \in C(u)} \min\{dp_0[c'], dp_1[c'], dp_2[c']\}\}$$

$$= \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0[c], dp_1[c], dp_2[c]\} + dp_2[u]\}$$

$$= \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0[c], dp_1[c], dp_2[c]\}\} + dp_2[u]$$

$$= \gamma + dp_2[u]$$

$$where\ \gamma = \min_{c \in C(u)} \{\min\{dp_1[c], dp_2[c]\} - min\{dp_0[c], dp_1[c], dp_2[c]\}\}$$

□

Algorithm 6.1 TRDF-trees

Input A weighted tree T fixed at a root r
Output total Roman domination number γ_{tR} of the weighted T

function TRD(T,u)
  $dp_0[u] \leftarrow 0$
  $dp_0^2[u] \leftarrow \infty$
  $dp_1[u] \leftarrow 1$
  $dp_1^{1,2}[u] \leftarrow \infty$
  $dp_2[u] \leftarrow 2$
  $dp_2^{1,2}[u] \leftarrow \infty$
  $\alpha \leftarrow \infty$
  $\beta \leftarrow \infty$
  $\gamma \leftarrow \infty$
  for each $c \in C(u)$ **do** ▷ DFS traversal of T
    TRD(T,c)
    $dp_0[u] \leftarrow dp_0[u] + \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\}$
    $\alpha \leftarrow \min\{\alpha, dp_2^{1,2}[c] - \min\{dp_0^2[c], dp_1^{1,2}[c], dp_2^{1,2}[c]\}$
    $dp_1[u] \leftarrow dp_1[u] + \min\{dp_0^2[c], dp_1[c], dp_2[c]\}$
    $\beta \leftarrow \min\{\beta,\ \min\{dp_1[c], dp_2[c]\} - min\{dp_0^2[c], dp_1[c], dp_2[c]\}\}$
    $dp_2[u] \leftarrow dp_2[u] + \min\{dp_0[c], dp_1[c], dp_2[c]\}$
    $\gamma \leftarrow \min\{\gamma,\ \min\{dp_1[c], dp_2[c]\} - min\{dp_0[c], dp_1[c], dp_2[c]\}\}$
  end for
  if u is non-leaf **then** ▷ All nodes other than leaf
    $dp_0^2[u] \leftarrow dp_0[u] + \alpha$
    $dp_1^{1,2}[u] \leftarrow dp_1[u] + \beta$
    $dp_2^{1,2}[u] \leftarrow dp_2[u] + \gamma$
  end if
end function

function PRINT-TRD(T,r) ▷ Let r be the root of the tree T
  TRD(T,r)
  return $\min\{dp_0^2[r], dp_1^{1,2}[r], dp_2^{1,2}[r]\}$
end function

6.2.2 Time and Space Complexity Analysis

Lemma 6.2.4. *For a tree T with n vertices, the time and space complexities of Algorithm 6.1 are $O(n)$ and $O(n)$, respectively.*

Proof. The time complexity of Algorithm 6.1 is $O(n)$ as we are using the DFS method to traverse the tree, which takes $O(n)$ time. We visit each vertex exactly once in the algorithm, and as the number of vertices is n, the total time complexity is $O(n)$. The space complexity is also $O(n)$. We create *six* arrays, $dp_0, dp_0^2, dp_1, dp_1^{1,2}, dp_2, dp_2^{1,2}$ of size n each, and *three* variables, α, β, γ. □

6.2.3 Proof of Correctness

Lemma 6.2.5. *Algorithm 6.1 computes the minimum total Roman domination number of a given tree $T = (V, E)$.*

Proof. We use induction to prove the correctness of the algorithm. We show that Algorithm 6.1 computes the *total Roman domination number* correctly for all trees of height $h \in \mathbb{Z}^+$.

Base Case: For $h = 0$, the tree contains a single node. Let us denote it by u. Now, $dp_0[u] = 0$, $dp_0^2[u] = \infty$, $dp_1[u] = 1$, $dp_1^{1,2}[u] = \infty$, $dp_2[u] = 2$, $dp_2^{1,2}[u] = \infty$,
$\gamma_{tR} = \min\{dp_0^2[u], dp_1^{1,2}[u], dp_2^{1,2}[u]\} = \min\{\infty, \infty, \infty\} = \infty$.

Induction Hypothesis: Let the correctness of the algorithm hold on trees of height $h'(< h)$.

Induction Step: Let T be a tree of height h and let us fix a vertex r to be the root of T. For all $c \in C(r)$, the height of the subtree rooted at c, $h' < h$. So, we have optimal values of $dp_0[c], dp_0^2[c], dp_1[c], dp_1^{1,2}[c], dp_2[c], dp_2^{1,2}[c]$ for all $c \in C(r)$. Using these and the results from Theorem 6.2.1, we can find the optimal values of $dp_0[r]$, $dp_0^2[r]$, $dp_1[r]$, $dp_1^{1,2}[r]$, $dp_2[r]$, $dp_2^{1,2}[r]$.

Now, the TRDN of the tree is $\gamma_{tR}(T) = \min\{dp_0^2[r], dp_1^{1,2}[r], dp_2^{1,2}[r]\}$. This is because, from the TRDF, the vertex r can be assigned either 0 or 1 or 2. If $f(r) = 0$, it is necessary that for at least one child $c \in C(r)$, $f(c) = 2$ to satisfy the first condition of total Roman domination, if $f(r) = 1$ or $f(r) = 2$, it is necessary that for at least one child $c \in C(r)$, $f(c) \in \{1, 2\}$ to satisfy the total Roman domination second condition, as r doesn't have any parent vertex. Hence, the induction hypothesis is true for all trees of height h.

Hence, we see that Algorithm 6.1 computes the total Roman domination number correctly for all trees of height $h \in \mathbb{Z}^+$. □

6.3 Conclusion

In this chapter, we proposed a linear time algorithm for the TRDS problem in trees. The space complexity of the proposed algorithm is also linear.

Chapter 7

Conclusions and Future Perspectives

In this thesis, we explored several variants of the domination problem within specific graph classes. These include the following.

(i) The *semi-total dominating set* problem in general graphs and UDGs.

(ii) The *total dominating set* problem in unit disk graphs and grid graphs.

(iii) The *total Roman dominating set* problem in UDGs and trees.

For the *semi-total dominating set* problem, we are the first to establish that it is NP-complete in UDGs. Next, we presented a 6-factor approximation algorithm for the T2DS problem in unit disk graphs. The running time of the algorithm is $O(nk)$, where n is the size of the vertex set and k is the size of a maximal independent set of the UDG. Using a similar technique, we improve the approximation factor to $\frac{53}{9}$. However, this enhancement increases the time complexity of the algorithm to $O(n^2)$. In addition, we also proposed a $2 + \ln(\Delta + 1)$-factor approximation algorithm for general graphs, where Δ is the degree of the given graph. We also observed that the semi-total domination problem in UDGs admits a PTAS with approximation factor $(2 + \epsilon)$ in time $n^{O(\frac{1}{\epsilon}\frac{1}{\log \epsilon})}$.

For the *total dominating set* problem, we proposed two approximation algorithms for UDGs with approximation factors 7.79 and 6.29 with running times $O(n^2)$ and $O(n^2m)$, respec-

tively, where n and m are the sizes of the vertex set and edge set. We have also shown that the *total dominating set* problem is NP-complete in grid graphs, a subclass of UDG.

Finally, We have delved into the *total Roman dominating set* problem and demonstrated its NP-completeness in unit disk graphs, like other domination variants. Initially, we devised a 10.5-factor approximation algorithm, followed by a refinement to achieve a 6.15-factor approximation algorithm with time complexities $O(n \log k)$ and $O(n^2 m)$, respectively. Additionally, we introduced a linear time dynamic algorithm designed specifically for solving the TRDS problem in trees.

Some of the intriguing areas for further investigation in the near future are as follows.

- designing new algorithms for the aforesaid problems with better approximation factors and time complexities.
- investigating the previously mentioned challenges in different graph classes such as in convex bipartite graphs, strongly chordal graphs etc.
- exploring the aforementioned challenges within alternative geometric constructs, such as rectangles, convex shapes, and non-convex structures.
- studying other variants of *dominating set* in the context of UDGs.
- inspecting other variants of the *Roman dominating set* problem in general graphs as well as in unit disk graphs.

Publications from the Contents of the Thesis

Papers published/submitted in international journals:

[J1] Sasmita Rout, Pawan Kumar Mishra, and Gautam Kumar Das, *Total Roman Domination and Total Domination in Unit Disk Graphs*, Communications in Combinatorics and Optimization (CCO), 2024 (in press).

[J2] Sasmita Rout and Gautam Kumar Das, *Improved Total Domination and Total Roman Domination in Unit Disk Graphs*, Discrete Mathematics, Algorithms and Applications (DMAA), 2023 (Submitted).

[J3] Sasmita Rout and Gautam Kumar Das, *Semi-total Domination in Unit Disk Graphs and General Graphs*, Discrete Applied Mathematics (DAM), 2024 (Submitted).

[J4] Sasmita Rout and Gautam Kumar Das, *A few Results on Total Domination and Total Roman Domination*, (Draft).

Papers published in international conference proceedings:

[C1] Sasmita Rout and Gautam Kumar Das, *Semi-total Domination in Unit Disk Graphs*, Conference on Algorithms and Discrete Applied Mathematics (CALDAM), Lecture Notes in Computer Science, pp. 117-129, 2024.

Bibliography

[1] H. Abdollahzadeh Ahangar, M. A. Henning, C. Löwenstein, Y. Zhao, and V. Samodivkin. Signed roman domination in graphs. *Journal of Combinatorial Optimization*, 27(2):241–255, 2014. [Pg.6], [Pg.17]

[2] H. A. Ahangar, M. P. Alvarez, M. Chellali, S. M. Sheikholeslami, and J. C. Valenzuela-Tripodoro. Triple roman domination in graphs. *Applied Mathematics and Computation*, 391:125444, 2021. [Pg.17]

[3] H. A. Ahangar, M. A. Henning, V. Samodivkin, and I. G. Yero. Total roman domination in graphs. *Applicable Analysis and Discrete Mathematics*, 10(2):501–517, 2016. [Pg.17], [Pg.67]

[4] K. M. Alzoubi, P.-J. Wan, and O. Frieder. Distributed heuristics for connected dominating sets in wireless ad hoc networks. *Journal of Communications and Networks*, 4(1):22–29, 2002. [Pg.2]

[5] J. Amjadi, S. Nazari-Moghaddam, S. Sheikholeslami, and L. Volkmann. Total roman domination number of trees. *Australasian. J. Combinatorics*, 69(2):271–285, 2017. [Pg.18]

[6] D. Amos and E. DeLaVina. On total domination in graphs. *Senior Project, University of Houston Downtown*, 2012. [Pg.14], [Pg.15]

BIBLIOGRAPHY

[7] S. Arumugam, K. Ebadi, and M. Manrique. Co-secure and secure domination in graphs. *Util. Math*, 94:167–182, 2014. [Pg.13]

[8] J. Asplund, R. Davila, and E. Krop. A vizing-type result for semi-total domination. *Discrete Applied Mathematics*, 258:8–12, 2019. [Pg.16]

[9] M. Atapour and N. Soltankhah. On total dominating sets in graphs. *arXiv preprint arXiv:0810.4667*, 2008. [Pg.13]

[10] S. Banerjee, J. M. Keil, and D. Pradhan. Perfect roman domination in graphs. *Theoretical Computer Science*, 796:1–21, 2019. [Pg.3]

[11] D. W. Bange, A. E. Barkauskas, L. H. Host, and P. J. . Generalized domination and efficient domination in graphs. *Discrete Mathematics*, 159(1-3):1–11, 1996. [Pg.13]

[12] C. Berge. *La theorie des graphes*. na, 1958. [Pg.1]

[13] C. Berge. *The Theory of Graphs and Its Applications*. Wiley, 1958. [Pg.11]

[14] T. Biedl and G. Kant. A better heuristic for orthogonal graph drawings. *Computational Geometry*, 9(3):159–180, 1998. [Pg.24]

[15] N. Bourgeois, F. Della Croce, B. Escoffier, and V. T. Paschos. Fast algorithms for min independent dominating set. *Discrete Applied Mathematics*, 161(4-5):558–572, 2013. [Pg.13]

[16] A. Brandstadt. On the linear structure and clique-width of bipartite permutation graphs. *Ars Combinatoria*, 67:273–281, 2003. [Pg.12]

[17] A. Cabrera Martinez, S. Cabrera Garcia, and A. Carrion Garcia. Further results on the total roman domination in graphs. *Mathematics*, 8(3):349, 2020. [Pg.66]

[18] A. Cabrera Martínez, L. P. Montejano, and J. A. Rodríguez-Velázquez. Total weak roman domination in graphs. *Symmetry*, 11(6):831, 2019. [Pg.18]

[19] A. Cabrera-Martínez and J. A. Rodríguez-Velázquez. A note on double domination in graphs. *Discrete Applied Mathematics*, 300:107–111, 2021. [Pg.3]

[20] E. W. Chambers, B. Kinnersley, N. Prince, and D. B. West. Extremal problems for roman domination. *SIAM Journal on Discrete Mathematics*, 23(3):1575–1586, 2009. [Pg.6]

[21] M. Chellali, O. Favaron, A. Hansberg, and L. Volkmann. k-domination and k-independence in graphs: A survey. *Graphs and Combinatorics*, 28(1):1–55, 2012. [Pg.13]

[22] M. Chellali and T. W. Haynes. A note on the total domination number of a tree. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 58:189–193, 2006. [Pg.14]

[23] M. Chellali, N. Jafari Rad, S. M. Sheikholeslami, and L. Volkmann. Varieties of roman domination ii. *AKCE International Journal of Graphs and Combinatorics*, 17(3):966–984, 2020. [Pg.6]

[24] Q. Chen and Y. Tang. Semitotal domination subdivision numbers of graphs. *Journal of Discrete Mathematical Sciences and Cryptography*, 23(5):973–987, 2020. [Pg.16]

[25] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979. [Pg.12]

[26] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete mathematics*, 86(1-3):165–177, 1990. [Pg.12], [Pg.71]

[27] E. J. Cockayne, R. Dawes, and S. T. Hedetniemi. Total domination in graphs. *Networks*, 10(3):211–219, 1980. [Pg.3], [Pg.13], [Pg.15]

[28] E. J. Cockayne, P. A. Dreyer Jr, S. M. Hedetniemi, and S. T. Hedetniemi. Roman domination in graphs. *Discrete mathematics*, 278(1-3):11–22, 2004. [Pg.5], [Pg.6], [Pg.13], [Pg.17]

[29] E. J. Cockayne, M. A. Henning, and C. M. Mynhardt. Vertices contained in all or in no minimum total dominating set of a tree. *Discrete Mathematics*, 260(1-3):37–44, 2003. [Pg.14]

[30] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT press, 2022. [Pg.44]

[31] G. D. da Fonseca, C. M. de Figueiredo, V. G. P. de Sá, and R. C. Machado. Efficient sub-5 approximations for minimum dominating sets in unit disk graphs. *Theoretical Computer Science*, 540:70–81, 2014. [Pg.12], [Pg.21], [Pg.22]

[32] P. Damaschke, H. Müller, and D. Kratsch. Domination in convex and chordal bipartite graphs. *Information Processing Letters*, 36(5):231–236, 1990. [Pg.12], [Pg.14]

[33] M. Damian-Iordache and S. V. Pemmaraju. A (2+ ε)-approximation scheme for minimum domination on circle graphs. *Journal of algorithms*, 42(2):255–276, 2002. [Pg.12]

[34] E. DeLaVina, Q. Liu, R. Pepper, B. Waller, and D. B. West. Some conjectures of graffiti. pc on total domination. *Congressus Numerantium*, 185:81, 2007. [Pg.14]

[35] M. Denny and S. Gaines. *Chance in biology: using probability to explore nature*. Princeton University Press, 2011. [Pg.6]

[36] P. Dorbec, M. A. Henning, and J. McCoy. Upper total domination versus upper paired-domination. *Quaestiones Mathematicae*, 30(1):1–12, 2007. [Pg.13]

[37] M. Dorfling, W. Goddard, and M. A. Henning. Domination in planar graphs with small diameter ii. *Ars Combinatoria*, 78:237–255, 2006. [Pg.14]

[38] P. A. Dreyer Jr. *Applications and variations of domination in graphs*. Rutgers The State University of New Jersey, School of Graduate Studies, 2000. [Pg.6], [Pg.17]

[39] D. Easley and J. Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*, volume 1. Cambridge university press Cambridge, 2010. [Pg.3]

[40] O. Favaron, H. Karami, R. Khoeilar, and S. M. Sheikholeslami. On the roman domination number of a graph. *Discrete Mathematics*, 309(10):3447–3451, 2009. [Pg.6], [Pg.17]

[41] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998. [Pg.6]

[42] F. V. Fomin, D. Kratsch, and G. Woeginger. *Graph-Theoretic Concepts in Computer Science*. Springer, 2006. [Pg.4]

[43] E. Galby, A. Munaro, and B. Ries. Semitotal domination: new hardness results and a polynomial-time algorithm for graphs of bounded mim-width. *Theoretical Computer Science*, 814:28–48, 2020. [Pg.16]

[44] W. Goddard, M. A. Henning, and C. A. McPillan. Semitotal domination in graphs. *Utilitas Mathematica*, 94, 2014. [Pg.15], [Pg.22], [Pg.43]

[45] J. L. Gross, J. Yellen, and M. Anderson. *Graph Theory and Its Applications*. Chapman and Hall/CRC, 2018. [Pg.3]

[46] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20:374–387, 1998. [Pg.2]

[47] G. Hao, L. Volkmann, and D. A. Mojdeh. Total double roman domination in graphs. *Communications in Combinatorics and Optimization*, 5(1):27–39, 2020. [Pg.18]

[48] F. Harary and T. W. Haynes. Nordhaus-gaddum inequalities for domination in graphs. *Discrete Mathematics*, 155(1-3):99–105, 1996. [Pg.3]

[49] J. Hartmanis. Computers and intractability: A guide to the theory of np-completeness (m. r. garey and d. s. johnson). *Siam Review*, 24(1):90, 1982. [Pg.1]

[50] T. W. Haynes and P. J. . Paired-domination in graphs. *Networks: An International Journal*, 32(3):199–206, 1998. [Pg.13]

BIBLIOGRAPHY

[51] T. W. Haynes, S. Hedetniemi, and P. . *Fundamentals of Domination in Graphs*. CRC press, 1998. [Pg.2], [Pg.3]

[52] T. W. Haynes, S. Hedetniemi, and P. . *Domination in Graphs: Volume 2: Advanced Topics*. Routledge, 2017. [Pg.3]

[53] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning. *Topics in Domination in Graphs*. Springer, 2020. [Pg.2]

[54] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning. *Domination in Graphs: Core Concepts*. Springer, 2023. [Pg.13]

[55] T. W. Haynes and M. A. Henning. Trees with unique minimum total dominating sets. *Discussiones Mathematicae Graph Theory*, 22(2):233–246, 2002. [Pg.14]

[56] T. W. Haynes and M. A. Henning. Trees with unique minimum semitotal dominating sets. *Graphs and Combinatorics*, 36(3):689–702, 2020. [Pg.16]

[57] S. T. Hedetniemi and R. C. Laskar. Bibliography on domination in graphs and some basic definitions of domination parameters. In *Annals of Discrete Mathematics*, volume 48, pages 257–277. Elsevier, 1991. [Pg.2], [Pg.3]

[58] S. T. Hedetniemi and R. C. Laskar. *Topics on Domination*. Elsevier, 1991. [Pg.3]

[59] M. A. Henning. A survey of selected recent results on total domination in graphs. *Discrete Mathematics*, 309(1):32–63, 2009. [Pg.2], [Pg.15]

[60] M. A. Henning. Total dominator colorings and total domination in graphs. *Graphs and Combinatorics*, 31:953–974, 2015. [Pg.15]

[61] M. A. Henning. Edge weighting functions on semitotal dominating sets. *Graphs and Combinatorics*, 33:403–417, 2017. [Pg.16]

[62] M. A. Henning and W. F. Klostermeyer. Italian domination in trees. *Discrete Applied Mathematics*, 217:557–564, 2017. [Pg.3]

[63] M. A. Henning, W. F. Klostermeyer, and G. MacGillivray. Perfect roman domination in trees. *Discrete Applied Mathematics*, 236:235–245, 2018. [Pg.3]

[64] M. A. Henning and A. J. Marcon. On matching and semitotal domination in graphs. *Discrete Mathematics*, 324:13–18, 2014. [Pg.15]

[65] M. A. Henning and A. J. Marcon. Semitotal domination in claw-free cubic graphs. *Annals of Combinatorics*, 20:799–813, 2016. [Pg.16]

[66] M. A. Henning and A. J. Marcon. Vertices contained in all or in no minimum semitotal dominating set of a tree. *Discussiones Mathematicae Graph Theory*, 36(1):71–93, 2016. [Pg.16]

[67] M. A. Henning and A. J. Marcon. Semitotal domination in graphs: Partition and algorithmic results. *Utilitas Mathematica*, 106, 2018. [Pg.16]

[68] M. A. Henning and V. Naicker. Disjunctive total domination in graphs. *Journal of Combinatorial Optimization*, 31:1090–1110, 2016. [Pg.15]

[69] M. A. Henning and A. Pandey. Algorithmic aspects of semitotal domination in graphs. *Theoretical Computer Science*, 766:46–57, 2019. [Pg.16], [Pg.36], [Pg.37]

[70] M. A. Henning and D. F. Rall. Trees with equal total domination and game total domination numbers. *Discrete Applied Mathematics*, 226:58–70, 2017. [Pg.15]

[71] M. A. Henning and A. Yeo. *Total Domination in Graphs*. Springer, 2013. [Pg.2], [Pg.15]

[72] S. K. Jena and G. K. Das. Total domination in geometric unit disk graphs. In *Proceedings of the 33rd Canadian Conference on Computational Geometry, 2021, August 10-12, 2021*, pages 219–227, 2021. [Pg.3], [Pg.14], [Pg.40], [Pg.41], [Pg.65]

[73] V. Kann. *On the approximability of NP-complete optimization problems*. PhD thesis, Royal Institute of Technology Stockholm, 1992. [Pg.12]

[74] Z. Kartal and A. Aytaç. Semitotal domination of harary graphs. *Tbilisi Mathematical Journal*, 13(3):11–17, 2020. [Pg.16]

[75] J. M. Keil. The complexity of domination problems in circle graphs. *Discrete Applied Mathematics*, 42(1):51–63, 1993. [Pg.12]

[76] W. Klostermeyer and C. Mynhardt. Secure domination and secure total domination in graphs. *Discussiones Mathematicae Graph Theory*, 28(2):267–284, 2008. [Pg.15]

[77] R. Laskar, J. Pfaff, S. M. Hedetniemi, and S. T. Hedetniemi. On the algorithmic complexity of total domination. *SIAM Journal on Algebraic Discrete Methods*, 5(3):420–425, 1984. [Pg.14]

[78] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982. [Pg.12], [Pg.22]

[79] M. Liedloff, T. Kloks, J. Liu, and S.-L. Peng. Efficient algorithms for roman domination on some classes of graphs. *Discrete Applied Mathematics*, 156(18):3400–3415, 2008. [Pg.6]

[80] C.-H. Liu and G. J. Chang. Roman domination on strongly chordal graphs. *Journal of Combinatorial Optimization*, 26(3):608–619, 2013. [Pg.6], [Pg.17]

[81] C.-H. Liu, S.-H. Poon, and J.-Y. Lin. Independent dominating set problem revisited. *Theoretical Computer Science*, 562:1–22, 2015. [Pg.13]

[82] M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25(2):59–68, 1995. [Pg.12], [Pg.21]

[83] A. J. Marcon. *Semitotal Domination in Graphs*. University of Johannesburg, 2015. [Pg.15]

[84] A. C. Martinez, A. M. Arias, and M. M. Castillo. A characterization relating domination, semitotal domination and total roman domination in trees. *Communications in Combinatorics & Optimization*, 6(2), 2021. [Pg.18]

[85] A. C. Martínez, S. C. García, and A. C. García. Further results on the total roman domination in graphs. *Distances and Domination in Graphs*, page 55, 2020. [Pg.17]

[86] A. C. Martínez, J. C. Hernández-Gómez, and J. M. Sigarreta. On the quasi-total roman domination number of graphs. *Mathematics*, 9(21):2823, 2021. [Pg.18]

[87] A. C. Martínez, D. Kuziak, and I. G. Yero. Outer-independent total roman domination in graphs. *Discrete Applied Mathematics*, 269:107–119, 2019. [Pg.18]

[88] I. Maza and A. Ollero. Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In *Distributed Autonomous Robotic Systems 6*, pages 221–230. Springer, 2007. [Pg.6]

[89] H. B. Merouane and M. Chellali. On secure domination in graphs. *Information Processing Letters*, 115(10):786–790, 2015. [Pg.13]

[90] D. A. Mojdeh and L. Volkmann. Roman {3}-domination (double italian domination). *Discrete Applied Mathematics*, 283:555–564, 2020. [Pg.18]

[91] T. Nieberg and J. Hurink. A ptas for the minimum dominating set problem in unit disk graphs. In *WAOA 2005, Palma de Mallorca, Spain, October 6-7, 2005, Revised Papers 3*, pages 296–306. Springer, 2006. [Pg.37]

[92] C. Padamutham and V. S. R. Palagiri. Algorithmic aspects of roman domination in graphs. *Journal of Applied Mathematics and Computing*, 64(1):89–102, 2020. [Pg.6]

[93] A. Pandey and B. Panda. Domination in some subclasses of bipartite graphs. *Discrete Applied Mathematics*, 252:51–66, 2019. [Pg.12]

[94] J. Pfaff, R. Laskar, and S. Hedetniemi. Np-completeness of total and connected domination and irredundance for bipartite graphs. technicial report 428, dept. math. *Sciences, Clemson University*, 1983. [Pg.14]

[95] A. Poureidi. Total roman domination for proper interval graphs. *Electronic Journal of Graph Theory and Applications*, 8(2):401–413, 2020. [Pg.2], [Pg.18]

[96] N. J. Rad and H. Rahbani. Some progress on the double roman domination in graphs. *Discussiones Mathematicae Graph Theory*, 39(1):41–53, 2019. [Pg.17]

[97] R. R. Rubalcaba and P. J. . Roman dominating influence parameters. *Discrete mathematics*, 307(24):3194–3200, 2007. [Pg.17]

[98] W. Shang, X. Wang, and X. Hu. Roman domination and its variants in unit disk graphs. *Discrete Mathematics, Algorithms and Applications*, 2(01):99–105, 2010. [Pg.65]

[99] Z. Shao, J. Amjadi, S. M. Sheikholeslami, and M. Valinavaz. On the total double roman domination. *IEEE Access*, 7:52035–52041, 2019. [Pg.6], [Pg.18]

[100] I. Stewart. Defend the roman empire! *Scientific American*, 281(6):136–138, 1999. [Pg.5], [Pg.17]

[101] B. L. Susada and R. G. Eballe. Independent semitotal domination in the join of graphs. *Asian Research Journal of Mathematics*, 19(3):25–31, 2023. [Pg.16]

[102] S. Thomassé and A. Yeo. Total domination of graphs and small transversals of hypergraphs. *Combinatorica*, 27(4):473–487, 2007. [Pg.14]

[103] L. G. Valiant. Universality considerations in vlsi circuits. *IEEE Transactions on Computers*, 100(2):135–140, 1981. [Pg.21], [Pg.56], [Pg.57]

[104] L. Volkmann. Signed total roman domination in graphs. *Journal of Combinatorial Optimization*, 32:855–871, 2016. [Pg.18]

[105] P.-J. Wan, K. M. Alzoubi, and O. Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. In *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1597–1604. IEEE, 2002. [Pg.65]

[106] D. B. West. *Introduction to Graph Theory*, volume 2. Prentice hall Upper Saddle River, 2001. [Pg.3]

[107] F. Xueliang, Y. Yuansheng, and J. Baoqi. Roman domination in regular graphs. *Discrete Mathematics*, 309(6):1528–1537, 2009. [Pg.6]

[108] J. Zhu. Approximation for minimum total dominating set. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, pages 119–124, 2009. [Pg.15]

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Department of Mathematics

Indian Institute of Technology Guwahati

Guwahati 781039, India