Approximation Algorithms for Dominating Set and its Variants on Unit Disk Graphs

A thesis submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

by

Ramesh Kumar Jallu

to the

Department of Mathematics

Indian Institute of Technology Guwahati, India

Guwahati-781 039, Assam, India

April 2018

Approximation Algorithms for Dominating Set and its Variants on Unit Disk Graphs

A thesis submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

by

Ramesh Kumar Jallu

(Roll No. 126123018)

Under the Supervision of

Dr. Gautam Kumar Das

to the

Department of Mathematics

Indian Institute of Technology Guwahati, India

Guwahati-781 039, Assam, India

April 2018

DECLARATION

It is certified that the work contained in the thesis entitled **"Approximation Algorithms for Dominating Set and its Variants on Unit Disk Graphs"** has been done by me, a student in the Department of Mathematics, Indian Institute of Technology Guwahati, India, under the guidance of **Dr. Gautam Kumar Das** for the award of Doctor of Philosophy and that this work has not been submitted elsewhere for a degree.

Place : IIT Guwahati

Date : April 9, 2018

Ramesh Kumar Jallu
Indian Institute of Technology Guwahati, India
Guwahati-781 039, Assam, India

CERTIFICATE

This is to certify that this thesis entitled "**Approximation Algorithms for Dominating Set and its Variants on Unit Disk Graphs**" being submitted by Mr. Ramesh Kumar Jallu to the Department of Mathematics, Indian Institute of Technology Guwahati, India, is a record of bona fide research work under my supervision and is worthy of consideration for the award of the degree of Doctor of Philosophy of the Institute.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree or diploma.

Place : IIT Guwahati
Date : April 9, 2018

Dr. Gautam Kumar Das
Associate Professor
Indian Institute of Technology Guwahati, India
Guwahati-781 039, Assam, India

Acknowledgements

Undertaking this PhD is a long journey and it would not have been possible to reach the destination without the support and guidance from many people. I would like to take this opportunity to express my deepest gratitude for their generous support.

Foremost, my heartfelt thanks to my thesis supervisor Dr. Gautam K. Das for allowing me to carry out the research work under his supervision and introducing me to the field of geometric approximation algorithms. This thesis is a result of his continuous support, guidance and encouragement for the past four and a half years at IIT Guwahati. I am grateful, and thank him for all the guidance he provided me, both in academic and personal contexts. I could not have imagined having a better advisor for my PhD.

Besides my advisor, I thank the members of my doctoral committee Prof. S. V. Rao, Dr. Partha Sarathi Mandal, and Dr. Sagarmoy Dutta for their suggestions to improve the thesis work.

I am also thankful to my co-authors, Prof. Subhas C. Nandy, Prof. Paz Carmi, Prof. Fonseca, Manjanna, Yael Stein, Sangram and Prajwal for their constant advices through e-mails. I also thank Dr. Subhabrata Paul for introducing us to one of the problems in this thesis.

My sincere thanks to the Ministry of Human Resource Development, Government of India, for providing me financial support for pursuing PhD at IIT Guwahati. I sincerely acknowledge IIT Guwahati for providing me the necessary facilities for conducting my research work smoothly. I also thank the Head of Department of Mathematics, IIT Guwahati, other teaching and non-teaching staff members of the Department for their support in administrative and technical works.

I would like to thank my friends and fellow research scholars, Anand, Subramani, and Swarup, with whom I spend most of my joyful time at IIT Guwahati. I acknowledge my seniors, Manjanna, Shibsankar, Barun, Govind, and juniors, Debashish and Sangram for their valuable suggestions. Special thanks to Manjanna for discussing the problems and explaining me many concepts.

Many thanks to my MTech friends Ramu, Aniket, Mahipal, and Murali, to name a few, who constantly motivated me to undergo for a PhD.

I express my gratitude to Dr. Anjan K. Chakrabarty, Prof. Sukanta Pati, and Dr. Inkulu, some of the dedicated and fantastic faculties of IIT Guwahati. I enjoyed their lectures during my course work, teaching assistance and enrich some of their teaching skills.

I must thank both Indian and abroad examiners for examining my thesis and providing their valuable suggestions.

Lastly, I am grateful to my parents for their support, encouragement, and unconditional love.

Ramesh Kumar Jallu

Abstract

Dominating set and its variants have been studied extensively in the literature and are of broad and current research interest to many researchers due to its wide range of applications, including, but not limited to, networks, VLSI, clustering, map labeling and coding theory. In this thesis, minimum dominating set problem and some of its variants, such as minimum connected dominating set, minimum liar's dominating set, and maximum (weighted) independent set problems are studied on *unit disk graphs*.

All the aforementioned problems are NP-hard on unit disk graphs. The high time complexity of the existing polynomial-time constant factor approximation algorithms motivated us to study these problems and design faster approximation algorithms and approximation schemes. The approximation algorithms and approximation schemes presented in this thesis outperform the existing algorithms in the literature in terms of time (space) complexity.

Firstly, we study the geometric minimum dominating set (GMDS) problem, defined as follows:

> *Given a set $\mathcal{P}$ of n points in $\mathbb{R}^2$, find a minimum cardinality subset $\mathcal{P}'$ of $\mathcal{P}$ such that each point in $\mathcal{P}$ lies inside the unit disk centered at some point in $\mathcal{P}'$.*

The GMDS problem is a restricted version of the minimum dominating set problem. We first present a simple $O(n \log k)$ time 5-factor approximation algorithm for the GMDS problem, where k is the size of the output. Next, we present 4-factor and 3-factor approximation algorithms in $O(n^6 \log n)$ and $O(n^{11} \log n)$ time, respectively, which improve the time complexities of the best known results by a factor of $O(n^2)$ and $O(n^4)$, respectively. We also present $\frac{14}{3}$-factor and $\frac{45}{13}$-factor approximation algorithms in time $O(n^5 \log n)$ and $O(n^{10} \log n)$, respectively. Finally, we propose a two-level shifting lemma. Using the two-level shifting lemma we present a $\frac{5}{2}$-factor approximation algorithm and a polynomial-time approximation scheme (PTAS) for the GMDS problem.

Secondly, we study the minimum connected dominating set (MCDS) problem on unit disk graphs, defined as follows:

> *Given a unit disk graph (UDG), $G = (V, E)$, find a minimum cardinality subset D of V satisfying, (i) D is a dominating set of G, and (ii) the subgraph induced by D is connected.*

We present a constant factor distributed approximation algorithm for the MCDS problem (in view of the given UDG represents a homogeneous wireless ad-hoc network), which runs in $O(\Delta)$ time, where Δ is the degree of the given UDG.

Next, we introduce a variant of dominating set problem in UDGs and we call this problem as the geometric minimum liar's dominating set (GMLDS) problem, defined as follows:

> *Given a set $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ of n points in $\mathbb{R}^2$, find a minimum size subset D of $\mathcal{P}$ satisfying the following two conditions: (i) for each point $p_i \in \mathcal{P}$ there exist at least two points in D which are at distance at most one from p_i, and (ii) for every distinct pair of points p_i and p_j in $\mathcal{P}$, D contains at least three points from the closed neighborhood union of p_i and p_j.*

We prove that the GMLDS problem is NP-hard. We propose $\frac{63}{2}$-factor, $\frac{732}{k}$-factor (for $3 \leq k \leq 183$), and $\frac{846}{k}$-factor (for $3 \leq k \leq 282$) approximation algorithms. The $\frac{63}{2}$-factor approximation algorithm runs in $O(n \log n)$, where as the running time of the other algorithms are $O(n^{k+1}\Delta)$ for their respective k. We also propose a PTAS, which runs in $n^{O(\frac{1}{\epsilon^2} \log \frac{1}{\epsilon})}$ time.

Finally, we study the geometric maximum independent set (GMIS) and geometric maximum weighted independent set (GMWIS) problems, defined as follows:

> *Given a set $\mathcal{P}$ of n points in $\mathbb{R}^2$, find a maximum cardinality subset $\mathcal{P}'$ of $\mathcal{P}$ such that the points in $\mathcal{P}'$ are pairwise independent. In the weighted version, we are given a weight function $w : \mathcal{P} \to \mathbb{R}^+$ and the objective is to find an independent set $\mathcal{P}'$ of $\mathcal{P}$ that maximizes the sum of the weights of its points.*

We propose a 2-factor approximation algorithm for the GMIS problem. Our algorithm runs in $O(n^2 \log n)$ time and uses $O(n^2)$ space. We also propose a PTAS for the same problem. The running time of our proposed PTAS is $n^{O(k)}$, where k is a positive integer. For the GMWIS problem, we propose 2.16-factor and 2-factor approximation algorithms. The proposed 2.16-factor approximation algorithm runs in $O(n \log^3 n)$ time and uses $O(n \log n)$ space. In the unweighted version of 2.16-factor approximation algorithm the complexities decrease to $O(n \log^2 n)$ time and uses $O(n)$ space. The proposed 2-factor approximation algorithm for the weighted version runs in $O(n^2 \log^2 n)$ time and $O(n^2 \log n)$ space. In the unweighted version of 2-factor approximation algorithm the complexities decrease to $O(n^2 \log n)$ time and $O(n^2)$ space.

Contents

List of Tables

List of Algorithms

List of Figures

List of Abbreviations

APX	The class of polynomial-time constant approximable problems
CDS	Connected dominating set
CLDS	Connected liar's dominating set
DS	Dominating set
DTIME	Deterministic time
DUDC	Discrete unit disk cover
FPTAS	Fully polynomial-time approximation scheme
GM(D)S	Geometric (minimum) dominating set
IS	Independent set
km**-CDS**	k-connected m-dominating set
LDS	Liar's dominating set
MANET	Mobile ad-hoc network
MCDS	Minimum connected dominating set
MDS	Minimum dominating set
MIS	Maximum independent set
MLDS	Minimum liar's dominating set
MWCDS	Minimum weighted connected dominating set
MWDS	Minimum weighted dominating set
NP	The class of non-deterministic polynomial-time solvable problems
NP-hard	The class of non-deterministic polynomial-time hard problems
OPT	Optimal solution
P	The class of deterministic polynomial-time solvable problems
PTAS	Polynomial-time approximation scheme
TLDS	Total liar's dominating set
UBG	Unit ball graph
UDG	Unit disk graph
VLSI	Very large scale integration
WANET	Wireless ad-hoc network

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 $\mathcal{P}$
$x(p)$	The x-coordinate of p
$y(p)$	The y-coordinate of p
$\overline{pq}$	The line segment joining p and q
$d(p, q)$	The Euclidean distance between p and q
$N[p]$	The closed neighborhood of a point p
$\delta(p)$	The unit disk centered at a point p
d	The diameter of a disk
$\Delta(S)$	The set of unit disks centered at points in S
$\delta(p) \cup \delta(q)$	The union region of the disks $\delta(p)$ and $\delta(q)$
$\delta(p) \cap \delta(q)$	The intersection region of the disks $\delta(p)$ and $\delta(q)$
w	The weight function from $\mathcal{P}$ to $\mathbb{R}^+$
μ_w	The median weight of points in a point set
$w_{\max}$	The maximum weight among the points in a point set
H	A horizontal strip
S_i	A subset of the points lying in H having certain properties
W_i	The sum of the weights of the points in S_i
w_{ret}	The weight of the point returned by a query
$\mathcal{R}$	A rectangular region in the plane
ℓ_v	A vertical line
ℓ_h	A horizontal line
$\forall$	For all
$\mid$	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 difference of $\mathcal{P}$ and $\mathcal{Q}$
$\lvert \cdot \rvert$	The cardinality of a set
k	A positive integer
χ	A cell of size $k \times k$
$\sum$	The addition of a sequence of numbers
$\square$	The end of a proof
$\emptyset$	The empty set
$\{\cdot\}$	The set notation
$G = (V, E)$	An undirected connected simple graph with vertex set V and edge set E
$e(\cdot, \cdot)$	An edge between two nodes in a graph
n	The cardinality of V (or) the cardinality of a point set $\mathcal{P}$
m	The cardinality of E (or) the number of points lying in a strip
$d_G(\cdot, \cdot)$	The number of edges on a shortest path between two vertices/sets in G
$deg(\cdot)$	The degree of a vertex
Δ	The maximum degree in G
$N^r(v)$	The r-th neighborhood of a vertex v
$\overline{G}$	The complement of G
$N_G[\cdot]$	The closed neighborhood of a vertex/set
$N_G(\cdot)$	The open neighborhood of a vertex/set
$N_G[A]$	The closed neighborhood of a set $A \subseteq V$
$K_{1,6}$	The star graph with 7 vertices
mis	The size of a maximal independent set
opt	The size of an optimal solution
ε	A positive real number less than 1
ρ	The value of $1 + \epsilon$
$\{\mathcal{A}_\varepsilon\}$	A collection of algorithms with ε as input parameter
$H(\cdot)$	The harmonic function
$T(n)$	The time complexity of an algorithm with input size n
$O(\cdot)$	The asymptotic big-oh notation
H_i	A non-empty cell with index i in the hexagonal partition of $\mathcal{R}$
$(\cdot, \cdot)$	A pair of nodes in a graph
$\lfloor x \rfloor$	The largest integer less than or equal to x
$\lceil x \rceil$	The smallest integer greater than or equal to x
$a \leftarrow b$	Variable a gets the value of b

Chapter 1

Introduction

Dominating set problem is one of the classical combinatorial optimization problems in Graph theory. Given an undirected graph $G = (V, E)$, a *dominating set* (DS) D is a subset of V such that every vertex in $V \setminus D$ is adjacent to at least one vertex in D. That is, each vertex $v \in V$ is either in D or there exists a vertex $u \in D$ such that $e(u, v) \in E$. A graph can have multiple dominating sets (see Figure 1.1). The vertex set V is a trivial dominating set for every graph $G = (V, E)$. The *dominating set problem* asks to find a DS of minimum size in a given graph. We call a DS of minimum cardinality as a *minimum dominating set* (MDS). The vertices in D are called as dominators and the rest are called as dominatees. A dominator dominates all its neighbors and itself.

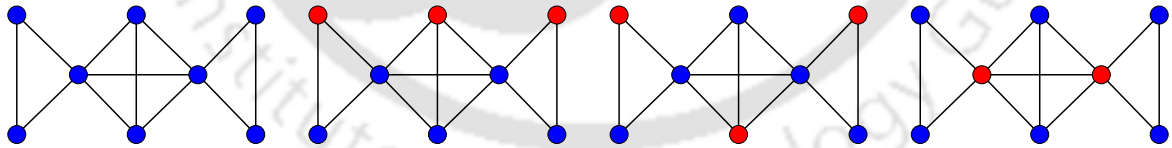


Figure 1.1: (a) An undirected graph, (b) a DS, (c) another DS, and (d) a CDS (the red vertices are dominating vertices).

The well-known applications of DS are in facility location, very large scale integration (VLSI), image processing, etc. In the facility location problem we are given a set of facilities (service providers) such as hospitals, fire stations, etc., and a set of clients. The

maximum distance that a client can travel to reach a facility is predefined. The objective is to open a subset of facilities such that every client is served by at least one facility. Opening a facility involves certain cost. Hence, we are interested in opening a minimum number of facilities such that every client is served by at least one opened facility within his/her predefined distance.

A *connected dominating set* (CDS) for a given connected undirected graph, $G = (V, E)$, is a dominating set and the subgraph induced by it is connected (see Figure 1.1(d)). The objective of the *minimum connected dominating set* (MCDS) problem is to find a CDS of minimum cardinality in G.

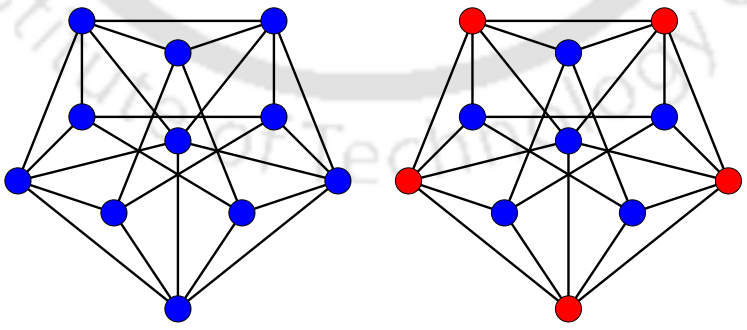
The widely used application of CDS is routing in wireless ad-hoc networks (WANETs) and mobile ad-hoc networks (MANETs). Unlike in wired networks, WANETs and MANETs do not have any centralized infrastructure such as routers, each node in the network participates in routing process. During message transfer from one node to another, the intermediate nodes act as relay nodes. This causes to reduce the network lifetime as the nodes are battery operated. One of the efficient ways to ensure multicast (one-to-many or many-to-many communication) in WANETs is by constructing a virtual backbone in the network. As we know WANETs do not have any physical backbone infrastructure, a virtual backbone can be constructed by the set of nodes in CDS [96]. In order to obtain a better performance of the network, a CDS of minimum size is desirable. The main idea behind the virtual backbone based routing is to improve the network performance such as network operational lifetime (as the nodes are battery operated), quality of service, etc. One can improve the lifetime of the network by selecting a minimum number of nodes such that communication between any two nodes can be established via these chosen nodes (i.e., only the chosen nodes keep routing information). Thus, the dominators (the chosen nodes) act as routers, whereas the non-dominator nodes use the services provided by the dominators.

In MANETs, the network topology gets changed frequently because of the mobility of nodes. The traditional routing protocols do not work in these kind of networks. The

best way to ensure routing in such situation is by forming clusters (groups of nodes) and electing a head for each cluster. Cluster heads take the role of routers in data transmission between nodes. A node moves from one cluster to another due to its mobility, in this case the node can communicate with the other nodes via its new cluster head. An effective routing can be achieved by choosing the minimum number of cluster heads. This will also ensure longer network operational time. The set of chosen cluster heads can be referred as a CDS.

A subset $D \subseteq V$ is a *k-tuple dominating set* in $G = (V, E)$, if each vertex $v \in V$ is dominated by at least k vertices in D. In other words, $|N_G[v] \cap D| \geq k$ for each $v \in V$, where $N_G[v]$ is the closed neighborhood of v in G, i.e., $N_G[v] = \{u \in V \mid e(u, v) \in E\} \cup \{v\}$. The minimum cardinality of a k-tuple dominating set of a graph G is called as the *k-tuple domination number of G.*

A subset $D \subseteq V$ is a *liar's dominating set* (LDS) if (i) for every $v \in V$, $|N_G[v] \cap D| \geq 2$ (i.e., D is a 2-tuple dominating set), and (ii) for every pair of distinct vertices u and v, $|(N_G[u] \cup N_G[v]) \cap D| \geq 3$ (see Figure 1.2). The liar's dominating set problem asks to find a liar's dominating set of minimum size in a given graph. We call an LDS of minimum cardinality as a minimum liar's dominating set (MLDS). The minimum cardinality of an LDS of a graph G is known as the liar's domination number of G. Every 3-tuple dominating set is a liar's dominating set, so the liar's domination number lies between 2-tuple and 3-tuple domination numbers.



(a) (b)

Figure 1.2: (a) An undirected graph G, and (b) a liar's dominating set of G (the set of red vertices).

The liar's dominating set problem is a variant of dominating set problem and gained researcher's attention due to its various important applications. Consider the following real-life scenario: we have an art gallery and our objective is to protect the paintings from an intruder such as a thief, or a saboteur, etc. Assume that the gallery has multiple entrances and each entrance is a possible location for an intruder. A protection device (a camera, sensor, etc.) placed at an entrance can not only detect (and report) an intruder entering through it but also at all the entrances that are visible from it. Now, the goal is to place protection devices as minimum as possible such that the intrusion of an intruder at any entrance is detected and reported. We can model the gallery as a graph. A vertex in the graph represents an entrance, and an edge corresponds to the respective entrances visibility one from the other. The goal can be achieved by finding a minimum dominating set (MDS) of the graph and placing the devices at all the vertices in the MDS. The protection devices are prone to failure. If any one of the devices placed is failed to detect the presence of the intruder, then to correctly identify and report the intruder, the devices must be placed at all the vertices of a minimum 2-tuple dominating set of the graph. Due to the transmission error, it may so happen that all the protection devices detect the intruder's location correctly but some of these devices can lie while reporting. Assume that at most one protection device in the closed neighborhood of the intruder can lie (misreport) at any point of time. In this scenario, in order to protect the gallery, we have to place the protection devices at the vertices of minimum liar's dominating set of the graph.

An *independent set* (IS) of a graph, $G = (V, E)$, is a set of vertices $V' \subseteq V$ such that no two vertices in V' are adjacent in G (see red vertices in Figure 1.1 (b), and (c)). The objective of the *independent set problem* for a given graph G is to find an independent set of maximum cardinality. We call an IS of maximum cardinality as a maximum independent set (MIS) or largest independent set of G.

The MIS problem has prime applications, including, but not limited to, map labeling, clustering, wireless networks (frequency assignment problem), and coding theory.

The following lemma describes the close relation between MDS and MIS.

Lemma 1.0.1. *[3] For any undirected graph G, $|MDS(G)| \leq |MIS(G)|$, where $|\cdot|$ denotes the cardinality of a set.*

In the weighted version of MIS, we are given a weight function $w : V \to \mathbb{R}^+$ and the objective is to find an independent set of maximum total weight. The weighted version of the MIS problem is known as maximum weighted independent set (MWIS) problem. The MIS problem is a special case of the MWIS problem with equal weights.

A *maximal independent set* of G is an independent set which is not a proper subset of any other independent set of G. An *independent dominating set* is a dominating set which is also an independent set. The following observation describes the relationship between maximal independent set and dominating set of a graph.

Observation 1.0.2. For any simple undirected graph G, every maximal independent set of G is a dominating set in G.

An algorithm for a minimization (resp. maximization) problem is said to be a *ρ-factor approximation algorithm* if for every instance of the problem the algorithm produces a feasible solution whose value is within a factor of ρ (resp. $\frac{1}{\rho}$) of the optimal solution value and runs in polynomial-time of the input size. Here, ρ is called as the approximation ratio or approximation factor of the algorithm.

A *polynomial-time approximation scheme* (PTAS) for an optimization problem is a collection of algorithms $\{\mathcal{A}_\varepsilon\}$ such that for a given $\varepsilon > 0$, $\mathcal{A}_\varepsilon$ is a $(1+\varepsilon)$-factor approximation algorithm in case of minimization problem ($(1-\varepsilon)$ in case of maximization). The running time of $\mathcal{A}_\varepsilon$ is required to be polynomial in the size of the problem depending on ε. If the running time of $\mathcal{A}_\varepsilon$ is polynomial in the size of the problem and in $\frac{1}{\varepsilon}$, then $\{\mathcal{A}_\varepsilon\}$ is said to be *fully polynomial-time approximation scheme* (FPTAS). Note that for a problem existence of FPTAS implies PTAS, but the converse need not be true.

The class APX is the set of problems in NP that admit constant factor approximation algorithms. A problem is said to be APX-hard if every problem in APX is reducible to

that problem via PTAS reduction. A problem is said to be APX-complete if it belongs to both APX and APX-hard.

An *intersection graph* of objects is a graph, where the vertex set is the set of objects and there is an edge between two objects if their intersection is nonempty. A *unit disk graph* (UDG) is an intersection graph of a family of disks of equal radii in the plane. Given a family $D = \{d_1, d_2, \ldots, d_n\}$ of n circular disks in the plane, each having radius 1, the corresponding UDG, $G = (V, E)$, is defined as follows: each vertex $v_i \in V$ corresponds to the center of the disk $d_i \in D$, and there is an edge between two vertices if and only if the Euclidean distance between the corresponding disk centers is at most 1, i.e., the corresponding disk centers lie one in the other disk (see Figure 1.3).

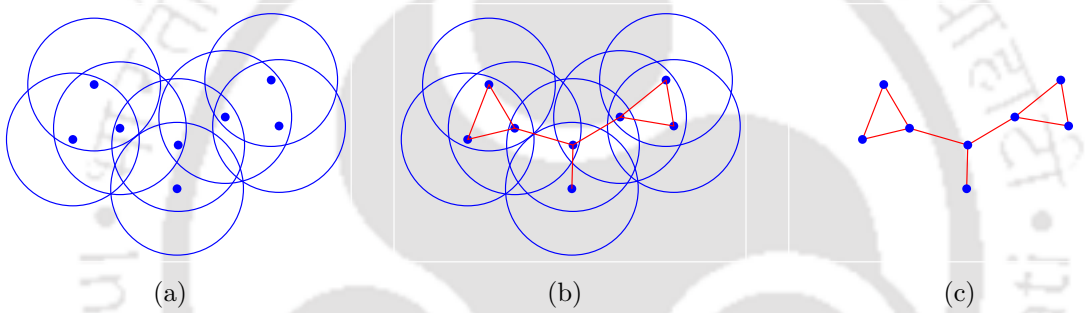


Figure 1.3: (a) A family of unit disks in the plane, (b) the corresponding unit disk graph, and (c) the final graph without disks.

Some interesting properties of UDGs are available in the literature [70, 101] and few of them are given below. These properties play vital role in bounding the approximation ratios.

1. Let d be a disk of radius r and let S be a set of disks of radius r such that every disk in S intersects d and no two disks in S intersect each other. Then, $|S| \leq 5$.

2. A UDG cannot have an induced subgraph isomorphic to $k_{1,6}$.

3. Let $G = (V, E)$ be a UDG and let $v \in V$ such that the unit disk corresponding to v has the smallest x coordinate. The size of a maximum independent set in $G(N_G(v))$ is at most 3, where $N_G(v)$ is the neighborhood of v.

1.1 Scope of the Thesis

The optimization problems such as, minimum dominating set (MDS) problem, minimum connected dominating set (MCDS) problem, minimum liar's dominating set (MLDS) problem, and maximum (weighted) independent set (MIS/MWIS) problem have many real-world applications. All these problems are NP-hard and none of them admit constant factor approximation algorithms for general graphs, unless $P = NP$. This motivated many researchers to study the problems for other graph classes. We study the problems for unit disk graphs with known disk position and focus on designing constant factor approximation algorithms and polynomial-time approximation schemes that outperform the existing results in the literature. We name these problems on unit disk graphs as the geometric versions of MDS, MCDS, MLDS, and MIS/MWIS problems in the Euclidean plane. In this thesis, we considered three variants of dominating set. Many other variants of dominating set and its applications can be found in the book *Fundamental of Domination in Graphs* by Haynes et al. [54].

1.2 Organization of the Thesis

The thesis contains seven chapters and is organized as follows.

Chapter 2: Literature Review. In this chapter, we discuss the existing literature related to the problems considered in this thesis.

Chapter 3: Geometric Minimum Dominating Set Problem. In this chapter, we start with the problem description and propose constant factor approximation algorithms including a PTAS for the geometric minimum dominating set (GMDS) problem. We compare the proposed results with the best-known results available in the literature.

Chapter 4: Distributed Minimum Connected Dominating Set problem. In this chapter, we propose a distributed algorithm to construct a connected dominating set in a given unit disk graph, which represents a wireless ad-hoc network.

We prove that the proposed algorithm runs in $O(\Delta)$ time and has $O(n)$ message complexity.

Chapter 5: Geometric Minimum Liar's Dominating Set Problem. In this chapter, we start with the problem description and study its hardness result. We propose constant factor approximation algorithms and a PTAS for the geometric minimum liar's dominating set (GMLDS) problem.

Chapter 6: Geometric Maximum (Weighted) Independent Set Problem. We start this chapter with the problem description, propose a 2-factor approximation algorithm, and a PTAS for the geometric maximum independent set (GMIS) problem. For the weighted case we propose 2.16-factor and 2-factor approximation algorithms.

Chapter 7: Conclusion and Future Work. In this chapter, we summarize the work done in this thesis and make concluding remarks. We also identify a number of open problems as future research.

Chapter 2

Literature Review

In this chapter, we discuss the state of the art results for all the considered problems, namely, geometric minimum dominating set (GMDS), minimum connected dominating set (MCDS), geometric minimum liar's dominating set (GMLDS) and GMIS/GMWIS, on general graphs, unit disk graphs, and other intersection graphs. The chapter is divided into four sections. In the first section, we define the GMDS problem and discuss the existing results of it. In Section 2.2, we define the MCDS problem and study its previous work available in the literature. In Section 2.3 and Section 2.4, we define the GMLDS and GMIS/GMWIS problems, respectively, and review latest results.

2.1 Minimum Dominating Set Problem

A dominating set of a simple undirected graph $G = (V, E)$ is a set of vertices $D \subseteq V$ such that every vertex $v \in V$, either (i) $v \in D$, or (ii) there exists $u \in D$ such that $e(u, v) \in E$. We call a dominating set (DS) of minimum cardinality as a minimum dominating set (MDS) and the problem of finding a dominating set of minimum cardinality as the *minimum dominating set problem*.

The MDS problem is known to be NP-hard for general graphs, and even for planar graphs [48]. A closely related problem to the MDS problem is the *set cover problem* in

the following sense. Given an instance of dominating set problem, we can construct an instance of the set cover problem and vice versa.

In the set cover problem, we are given a ground set $\mathcal{E} = \{e_1, e_2, \ldots, e_n\}$ of n elements and a collection $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of m subsets of $\mathcal{E}$ such that $\mathcal{E} = \bigcup_{i=1}^{m} S_i$. The objective is to find a smallest collection of $\mathcal{S}$ whose union is equal to $\mathcal{E}$.

Given a graph $G = (V, E)$ with vertex set $V = \{v_1, v_2, \ldots, v_n\}$, we can construct an instance $(\mathcal{E}, \mathcal{S})$ for the set cover problem as follows: the ground set $\mathcal{E} = V$ and $S_i = \{v_i\} \cup \{v_j \mid e(v_i, v_j) \in E\}$ for $i = 1, 2, \ldots, n$. Observe that (i) D $(\subseteq V)$ is a dominating set of G if and only if $\mathcal{S}' = \{S_i \mid v_i \in D\}$ is a feasible solution for the set cover problem, and (ii) $|D| = |\mathcal{S}'|$.

Let $(\mathcal{E}, \mathcal{S})$ be an instance of the set cover problem with the collection of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of the universe $\mathcal{E} = \{e_1, e_2, \ldots, e_n\}$. Now we construct an instance $G = (V, E)$ for the dominating set problem from the given instance of the set cover problem as follows : $V = \mathcal{E} \cup \mathcal{S}$ and the edge set $E = \{e(S_i, S_j) \mid S_i, S_j \in \mathcal{S}\} \cup \{e(e_i, S_j) \mid e_i \in \mathcal{E} \cap S_j\}$. Observe that a subset $\mathcal{C}$ of $\mathcal{S}$ is a feasible solution for the set cover problem if and only if $\mathcal{C}$ is a dominating set for G.

Therefore, an α-factor approximation algorithm for the dominating set problem implies an α-factor approximation algorithm for the set cover problem and vice versa. Raz and Safra [86] showed that the set cover problem is not approximable within a factor of $c \log n$, for some $c > 0$, unless $P = NP$, where n is the size of the input. So the dominating set problem is also not approximable within a factor of $c \log n$, for some $c > 0$, unless $P = NP$.

As it is not possible to have constant factor approximation algorithms for the MDS problem in general graphs unless $P = NP$, researchers considered the same problem for special graphs.

The MDS problem is NP-hard on UDGs [27]. Unlike in the general graphs, constant factor approximation algorithms exist for the MDS problem on UDGs in the literature. Marathe et al. [70] proposed a linear time 5-factor approximation algorithm by finding

a maximal independent set and using the property that UDGs cannot have $K_{1,6}$ as its induced subgraph. Most of the approximation algorithms in the literature assume that the geometric representation (i.e., coordinates) of the disk centers is given. The assumption of the geometric representation of the disks centers allows to partition the plane into grids and solve each grid separately, which leads to an approximation result to the problem. De et al. [34] considered a square partition of the plane. The problem of finding the geometric representation for a given UDG is NP-hard. Because, if we can have a polynomial-time algorithm to find the geometric representation for a given UDG, we can use the same algorithm to verify any given graph is UDG or not, but the latter problem is NP-hard [14].

Hunt III et al. [58] proposed a PTAS for the MDS problem on UDGs by using the shifting lemma similar to [55]. The algorithm runs in $n^{O(k^2)}$, where k is the smallest integer such that $(\frac{k+1}{k})^2 \leq 1+\varepsilon$, for a given $\varepsilon > 0$. The 5-factor approximation algorithm proposed by Marathe *et al.* [70] and PTAS proposed by Nieberg and Hurink [76] do not require the geometric representation of the UDG. A very few approximation algorithms available in the literature in case of independence geometric UDG representation. Nieberg and Hurink [76] introduced the concept of 2-separated collection[1] of subsets in order to obtain a PTAS. The basic idea of the algorithm is as follows: start with an arbitrary vertex v and expand its neighborhood as long as $|D(N^{r+2}(v))| > (1+\varepsilon)|D(N^r(v))|$ holds, where $D(N^r(v))$ is an MDS of r-th neighborhood of v. Eliminate $N^{\hat{r}+2}(v)$ from the graph, where $\hat{r}$ is the smallest r violating the inequality, and continue the same process for the remaining graph. The union of the MDSs obtained for $(\hat{r}+2)$-nd neighborhoods in all the iterations is a DS for the whole instance. The collection of $\hat{r}$-th neighborhoods, i.e, $N^{\hat{r}}(v)$, obtained in all the iterations is a 2-separated collection. The running time of the PTAS is $O(n^{c^2})$, where $c = O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$.

[1]For a graph $G = (V, E)$, let $S := \{S_1, S_2, \ldots, S_k\}$ be a collection such that $S_i \subseteq V$, $i = 1, 2, \ldots, k$. The set S referred to be as an *m-separated collection*, if $d_G(u, v) > m$ for any two arbitrary vertices $u \in S_i$ and $v \in S_j$ with $i \neq j$, where $d_G(u, v)$ denotes the number of edges on a shortest path between u and v in G.

Fonseca et al. [42] considered the case where only the adjacency information of the UDG is given instead of the positions of the disks and proposed $\frac{44}{9}$-factor and $\frac{43}{9}$-factor approximation algorithms that run in $O(n+m)$ and $O(n^2m)$ time, respectively, where n is the number of nodes and m is the number of edges of the graph. If the geometry of the graph is given the former algorithm runs in $O(n \log n)$ (regardless of m) and is based on local improvements to the independent dominating set obtained in [70]. De at al. [34] proposed 12-factor, 4-factor, and 3-factor approximation algorithms for the GMDS problem with running time $O(n \log n)$, $O(n^8 \log n)$, and $O(n^{15} \log n)$, respectively. They also proposed a PTAS which has performance ratio $(1+\frac{1}{k})^2$ in $n^{O(k)}$ time, where k is a positive integer.

Ambühl et al. [5] proposed a 72-factor approximation algorithm for the minimum weight dominating set (MWDS) problem on UDGs, where each vertex has a positive weight and the objective is to find a dominating set of total minimum weight. Observe that the MDS problem is a special case of the MWDS problem and hence all the results available for the MWDS problem in the literature are also applicable to the MDS problem. The basic idea of Ambühl et al. is as follows: (i) partition the plane into squares of side length μ $(= 0.999)$, (ii) solve each non-empty square (i.e., a square that contains at least one disk center) independently, and (iii) report the union of the solutions of the sub-problems. Huang et al. [57], Dai and Yu [29], and Zou et al. [104] improved the approximation factor for the MWDS problem to $6+\varepsilon$, $5+\varepsilon$, and $4+\varepsilon$, respectively. The strategy is similar in all the three algorithms. First, solve a sub-problem with approximation ratio α $(\alpha = 6, 5, 4)$ and use this result to obtain a $(\alpha+\varepsilon)$-factor approximation algorithm for the original problem. The running time of the algorithm highly depends on the number of times the sub-problem is invoked and is a very high degree polynomial function in n. For polynomial growth bounded graphs[2], Wang et al. [100] presented a

[2]A graph G is said to be *polynomial growth bounded graph* if there exists polynomial function $f(\cdot)$ such that the r-th neighborhood of every vertex in G contains at most $f(r)$ independent vertices, where $f(r) = O(r^c)$ for some constant $c \geq 1$. The function $f(\cdot)$ does not depend on the number of independent vertices, but on r. If r is constant, the number of independent vertices in an r-neighborhood of a vertex is bounded by a constant. Unit disk graphs and r-regular polygons of unit size, for example, are special

PTAS for the MWDS problem using the local neighborhood-based scheme introduced by Nieberg et al. [78] and the PTAS runs in $n^{O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})}$ time.

Carmi et al. [19] studied the MDS problem on disk graphs[3] and proposed a 5-factor approximation algorithm. A local search based PTAS on disk graphs proposed by Gibson and Pirwani [49], which runs in $n^{O(\frac{1}{\varepsilon^2})}$ time. For intersection graphs of arbitrary fat objects the problem is known to be as hard to approximate as for general graphs and for arbitrary rectangles, the problem is known to be APX-hard, due to Erlebach and van Leeuwen [39]. They also proposed constant factor approximation algorithms with the aid of ε-net [52] for various geometric intersection graphs of objects such as disks of constant ply, r-regular polygons (r is an arbitrary constant), pairwise homothetic triangles, and rectangles with bounded aspect ratio. In the case of arbitrary fat objects with bounded ply, they get a $(3+\varepsilon)$-factor approximation algorithm. For the intersection graph of restricted set of r-regular polygons, Kammer and Tholey [61] proposed $O(1)$-factor approximation algorithms.

Unless $P = NP$, the MDS problem does not admit an FPTAS on intersection graphs, even on UDGs [39, 65]. MDS problem is known to be APX-hard on intersection graphs of d-dimensional axis parallel boxes for any $d \geq 3$ [26].

In the *discrete unit disk cover* (DUDC) problem, two sets of points, namely P and Q, are given in $\mathbb{R}^2$, and the objective is to choose a minimum number of unit disks D' centered at the points in Q such that the union of the disks in D' covers all the points in P. Johnson [60] proved that the DUDC problem is NP-hard. Mustafa and Ray [74] proposed a $(1+\delta)$-factor approximation algorithm (PTAS) for the DUDC problem, $0 < \delta \leq 2$, using ε-net based local improvement approach. The fastest algorithm for getting a 3-factor approximation result is obtained by putting $\delta = 2$, and it runs in $O(m^{65}n)$ time, where $m = |Q|$ and $n = |P|$ [33]. The high complexity of the PTAS motivated the researchers to get efficient constant factor approximation algorithms for the DUDC problem. A series of such results in the literature are listed in Table 2.1.

cases of polynomial growth bounded graphs.

[3]intersection graph of disks of arbitrary radii

Table 2.1: Existing results for the DUDC problem

Ratio	Time complexity	Reference
108	polynomial	Călinescu et al., 2004 [17]
72	polynomial	Narayanappa and Voytechovsky, 2006 [75]
38	$O(m^2n^4)$	Carmi et al., 2007 [18]
22	$O(m^2n^4)$	Claude et al., 2010 [28]
$(1+\varepsilon)$ for $0 < \varepsilon \leq 2$	$O(m^{2(\frac{8\sqrt{2}}{\varepsilon})^2+1}n)$	Mustafa and Ray, 2010 [74]
18	$O(mn + n\log n + m\log m)$	Das et al., 2012 [33]
15	$O(m^6n)$	Fraser and López-Ortiz, 2012 [43]
$(9+\varepsilon)$ for $0 < \varepsilon \leq 6$	$O(m^{3(1+\frac{6}{\varepsilon})}n\log n)$	Acharyya et al., 2015 [9]

The MDS problem is a restricted version of the DUDC problem, where $P = Q$. Thus, the MDS problem can be viewed as a covering problem, where the objective is to cover a set of points with the minimum number of unit radius disks centered at those points. Therefore, all results for the DUDC problem are applicable to the MDS problem. In general, the covering objects can be squares, rectangles, convex, and non-convex objects, etc. If the covering object is a convex polygon, for example, instead of a unit disk, we can use the algorithm in [8] by modifying according to our objective to get a feasible solution.

2.2 Minimum Connected Dominating Set Problem

Connected dominating set of a simple connected undirected graph $G = (V, E)$ is a set of vertices $D \subseteq V$ with the following two properties : (i) D is a dominating set of G, and (ii) the subgraph induced by D is connected. We call a connected dominating set (CDS) of minimum cardinality as minimum connected dominating set (MCDS) in G. The *minimum connected dominating set problem* asks to find a CDS of minimum cardinality in a given graph. The MCDS problem is known to be NP-hard for general graphs [48]. Guha and Khuller [50] presented two greedy heuristic approximation algorithms with performance ratio $2(H(\Delta) + 1)$ and $\ln \Delta + 3$, respectively, where Δ is the degree of the graph, and $H(\cdot)$ is the harmonic function. The authors also presented an approximation

preserving reduction from the set cover problem to MCDS problem, which implied that no polynomial-time algorithm can achieve approximation ratio $(1-\varepsilon)H(\Delta)$ for any fixed ε $(0 < \varepsilon < 1)$, unless $NP \subseteq DTIME[n^{O(\log \log n)}]$ [69].

The MCDS problem is well-studied problem due to its primary application in wireless ad-hoc networks (WANETs) and mobile ad-hoc networks (MANETs) [70]. A CDS can be used as a virtual backbone due to the non-existence of centralized administration and network infrastructure in WANETs and MANETs. A CDS in WANETs and MANETs plays a crucial role in routing, broadcasting, and other network management functions [31]. Thus, a CDS of minimum size is desired. The literature [13] is a good survey for this problem in ad-hoc networks. The network topology of a WANET can be modeled as a UDG [27], $G = (V, E)$, in which two nodes (devices) are connected if and only if one is in the range of the other.

The problem of finding an MCDS in UDG is known to be NP-hard [27]. Many distributed approximation and heuristic algorithms have been designed in the literature [4, 23, 44, 45, 46, 72, 73, 96, 97]. Most of the distributed schemes first construct a maximal independent set and then convert it to a connected set by adding some extra nodes. This approach has several advantages over centralized and other distributed algorithms [94]. The quality of the solution highly depends on two factors: (i) the size of the maximal independent set constructed, and (ii) number of extra vertices that are required to make the maximal independent set connected.

As the quality of the solution in maximal independent set based heuristics depends on the ratio of the size of a maximal independent set to the size of MCDS, several studies have been done to establish the relation between maximal independent set and MCDS. A detailed list is given in Table 2.2. We use $mis(G)$ and opt to denote the size of a maximal independent set and the size of an optimal solution, respectively, in a given UDG G.

Algorithms proposed by Wan et al. [96] and Cheng et al. [23] obtain a performance ratio of 8 and time complexity $O(n)$, where n represents the number of nodes in the

Table 2.2: Relation between a maximal independent set and an MCDS on UDGs

Relation	Reference
$mis(G) \leq 5opt$	Marathe et al. [70]
$mis(G) \leq 4opt + 1$	Wan et al. [96]
$mis(G) \leq 3.8opt + 1.5$	Wu et al. [101]
$mis(G) \leq 3.453opt + 8.291$	Funke et al. [44]
$mis(G) \leq 3.4306opt + 4.8185$	Li et al. [67]
$mis(G) \leq 3.399opt + 4.874$	Du et al. [37]

ad-hoc network. However, they have high message complexity of $O(n \log n)$. Moreover, these heuristics are based on construction of a spanning tree, which makes it very costly in terms of communication overhead to maintain the MCDS in case of mobility and topology changes [4]. Alzoubi et al. [4], Cheng et al. [23], and Gao et al. [45] proposed constant factor approximation algorithms, which run in linear time and have linear message complexity. Some algorithms [44, 72, 73] are Steiner tree based, i.e., they construct a Steiner tree in order to connect nodes of the MIS obtained. However, these algorithms either have high time complexity or message complexity. Table 2.3 summarizes the results of the major algorithms for MCDS in UDG. In the table, n and m represent the number of nodes and edges of the UDG, respectively.

Table 2.3: Comparison of well known CDS construction algorithms on UDGs

Reference	Bound	Time complexity	Message complexity
Alzoubi et al. [4], 2002	$192opt + 48$	$O(n)$	$O(n)$
Wan et al.[96], 2002	$8opt + 1$	$O(n)$	$O(n \log n)$
Wang et al. [97], 2005	$192opt$	$O(n)$	$O(n\Delta)$
Cheng et al. - I [23], 2006	$8opt$	$O(n)$	$O(n \log n)$
Cheng et al. -II [23], 2006	$147opt + 33$	$O(n)$	$O(n)$
Funke et al. [44], 2006	$6.91opt + 16.58$	$O(n \times opt)$	$O(n \times opt)$
Min et al. [72], 2006	$6.8opt$	$O(n + m)$	$O(n)$
Muhammad et al. [73], 2006	$10opt + 1$	$O(mn \log n)$	$O(n)$
Gao et al. [46], 2012	$7.186opt + 1.2$	$O(\Delta mis(G)$	$O(m)$
This thesis	$104opt + 52$	$O(\Delta)$	$O(n)$

Cheng et al. [24] presented a PTAS for the MCDS problem which runs in $n^{O((\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})^2)}$ time. Leeuwen [65] studied the problem on UDGs with bounded density and presented an asymptotic FPTAS. Extension of PTASs (available in the literature) to higher dimen-

sions is not possible as the PTASs fully depend on geometric properties which hold in the plane but are no longer true in the space [102]. The MCDS problem is also studied in higher dimensions. The graph associated with the points in higher dimension is known to be a unit ball graph (UBG). Butenko et al. [16] presented a 22-factor approximation algorithm for the MCDS problem on UBG. Their algorithm first finds a maximal independent set and then connects it. Zhang et al. [102] considered the problem in d-dimension and presented a PTAS, which runs in $n^{O(\frac{1}{\varepsilon^d})}$ time. The algorithm runs faster than Cheng et al. [24] algorithm for $d = 2$.

The MCDS problem has been studied extensively in homogeneous ad-hoc networks (i.e., all nodes have the same transmission range) based on UDG construction. However, in practice, the transmission ranges of all nodes are not necessarily equal. In this case, the network can be modeled as a disk graph in which the edges are either unidirectional or bidirectional. Du et al. [35] studied the problem for disk graphs where both unidirectional and bidirectional links are considered and proposed two constant factor approximation algorithms. Thai et al. [93] presented two approximation algorithms in disk graphs with only bidirectional links. Thai et al. [92] proposed a constant factor approximation algorithm for of unidirectional links. The performance ratio of the algorithms in [35, 92, 93] is constant only if the ratio of the maximum transmission range over the minimum transmission range in the network is bounded.

The minimum weighted connected dominating set (MWCDS) problem is a weighted version of the MCDS problem, where each vertex is associate with a positive weight and the objective is to find a CDS of minimum total weight. The best known approximation ratio for the MWCDS problem in general graphs is $O(\log n)$ as MCDS is a special case of the MWCDS problem and the MCDS problem cannot have approximation ratio better than $O(\log n)$, unless $NP = DTIME[n^{O(\log \log n)}]$.

The MWCDS problem is well studied on UDGs too. Wang and Li [99] presented distributed algorithms that achieve approximation ratio $O(\min\{\log \Delta, \nu\})$, where Δ is the maximum node degree and ν is the ratio of the maximum weight to the minimum weight

of a node. However, these approximation ratios are no better than the approximation ratios known for general graphs, in the worst case. Ambühl et al. [5] gave the first constant factor approximation algorithm with ratio 89 by giving a 72-factor approximation algorithm for the MWDS problem and by introducing a 17-factor approximation algorithm for the connecting part. Huang et al. [57] and Zou et al. [103] improved the approximation factor to $10 + \varepsilon$ and $8.875 + \varepsilon$, respectively. Zou et al. [104] further improved the approximation factor to $5+\varepsilon$ by proposing $(4+\varepsilon)$-factor and $(1+\varepsilon)$-factor approximation algorithms for the MWDS problem and connecting part, respectively.

Another variant of the CDS problem known as k-connected m-dominating set problem is available in the literature. A CDS acts as a virtual backbone in wireless networks. The nodes (devices) in a wireless network are prone to failure due to several reasons such as energy depletion, accidental damage, a link may fade away due to movement of the nodes (in case of MANETs), etc. Hence, it is desirable to have a fault-tolerant backbone. To achieve flexible routing in case of network failure it is important to maintain a certain degree of redundancy in the constructed CDS [66]. Thus, to ensure the robustness of the backbone there must be multiple paths between any two nodes in the CDS and also a node (not in CDS) must have multiple dominators in CDS. For a graph $G = (V, E)$, a k-connected m-dominating set (km-CDS in short) is a subset of the vertices with the following properties: (i) the graph induced by the subset is k-connected, i.e., between any two nodes in the subset there are at least k disjoint paths, and (ii) every vertex (not in the subset) is adjacent to at least m nodes in the subset. The k-connectivity ensures that the communication will not be disrupted even $k - 1$ paths fail. Note that the km-CDS problem is nothing but the CDS problem when $k = m = 1$.

Dai and Wu [30] proposed three probabilistic algorithms for $k = m$. Kim et al. [63] studied the problem for $k = 3$ and given a constant factor approximation algorithm. For $k = 2$ and $m = 1$, an augmentation based CDS algorithm is proposed by Wang et al. [98]. The algorithm contains two phases: the first phase constructs a CDS and the second phase adds extra nodes to make all the backbone nodes to be in the same

block. Shang et al. [88] designed three centralized algorithms. The first algorithm is for $k = 1$ and has performance ratio $(5 + \frac{5}{m})$ for $m \leq 5$ and 7 for $m > 5$. This algorithm first constructs a maximal independent set based CDS and then repeatedly adds $m - 1$ maximal independent sets to make the CDS an m-dominating set. The second algorithm is for $k = 2$ and has performance ratio $(5 + \frac{25}{m})$ for $2 \leq m \leq 5$ and 11 for $m > 5$. The basic idea of this algorithm is similar to the algorithm proposed by Wang et al. [98]. The last algorithm is for $3 \leq k \leq m$ which first constructs a k-connected k-dominating set and repeatedly adds $m - k$ maximal independent sets to make an m-dominating set. Li et al. [68] presented a distributed algorithm with performance ratio $(5 + \frac{5}{m})(k^2 + 1)$ for $m \leq 5$ and $7(k^2 + 1)$ for $m \geq 6$. Recently, Shi et al. [89] proposed constant factor approximation algorithms for the weighted version of the (k, m)-CDS problem, for $m \geq k$. Their algorithm first computes an m-dominating set. Then, it computes a k-connected subgraph containing the m-dominating set obtained in the first step. More results for the MCDS and MWCDS problems are available in [36].

2.3 Minimum Liar's Dominating Set Problem

A Liar's dominating set of a simple undirected graph $G = (V, E)$ is a dominating set L having the following two properties: (i) for every $v \in V$, $|N_G[v] \cap L| \geq 2$, and (ii) for every distinct pair of vertices u and v in V, $|(N_G[u] \cup N_G[v]) \cap L| \geq 3$, where $N_G[\cdot]$ is the closed neighborhood of a vertex. For a given graph G, the problem of finding a liar's dominating set (LDS) in G of minimum cardinality is known as the *minimum liar's dominating set problem* (MLDS).

The liar's domination problem is introduced by Slater in 2009. He showed that the problem is NP-hard for general graphs and given a lower bound on liar's domination number in case of tree graphs, by proving any liar's dominating set for a tree of order n contains at least $\frac{3}{4}(n + 1)$ vertices [90] and at most n. In his seminal work, Slater observed that there is subclass of trees for which the only LDS is the entire vertex set.

For example, the class of trees $\mathcal{T}$ such that for each vertex v in $\mathcal{T}$, either v is a leaf or at least one component of $\mathcal{T} \setminus \{v\}$ has cardinality at most two. Later, Roden and Slater [87] characterized tree classes with liar's domination number is equal to $\frac{3}{4}(n+1)$. In the same paper, they also showed that the MLDS problem is NP-hard even for bipartite graphs. Panda and Paul [81] proved that the problem is NP-hard for split graphs and chordal graphs. They also proposed a linear time algorithm for computing a minimum cardinality liar's dominating set in a tree.

Panda et al. [84] studied approximability of the problem and presented $O(\ln \Delta)$-factor approximation algorithm, where Δ is the degree of the graph. Panda and Paul [82] considered the problem for proper interval graphs and proposed a linear time algorithm for computing a minimum cardinality liar's dominating set. The problem is also studied for bounded degree graphs, and p-claw free graphs [84]. The MLDS problem is also studied on two-dimensional grid graphs and bounds on liar's domination are presented in [91].

Alimadadi et al. [2] provided the characterization of graphs and trees for which liar's domination number is $|V|$ and $|V|-1$, respectively. The authors observed that, a connected graph G with order (number of vertices) $n \geq 3$ has liar's domination number n if and only if every vertex v in G satisfies at least one of the following conditions: (i) $deg(v) = 1$, (ii) at least one component of $G \setminus \{v\}$ has order at most two , (iii) v belongs to an end-block (a block having at most one cut vertex of G) of order 3. Similarly, they characterized trees satisfying certain conditions have liar's domination number $n-1$. For connected graphs with girth (the length of a shortest cycle) at least five, they obtained an upper bound for the ratio between the liar's domination number and the 2-tuple domination number.

Panda and Paul [80, 83] studied variants of liar's dominating set, namely, connected liar's domination and total liar's domination. A *connected liar's dominating set* (CLDS) is an LDS whose induced subgraph is connected. A *total liar's dominating set* (TLDS) is dominating set L with the following two properties: (i) for every $v \in V$, $|N_G(v) \cap L| \geq 2$,

and (ii) for every distinct pair of vertices u and v, $|(N_G(u) \cup N_G(v)) \cap L| \geq 3$, where $N_G(\cdot)$ is the open neighborhood of a vertex. The objective of both the problems is to find CLDS and TLDS of minimum size, respectively. They proved that both the problems are NP-hard and proposed $O(\ln \Delta)$-factor approximation algorithms. They also proved that the problems are APX-complete for graphs with maximum degree 4.

We are the first to introduce the MLDS problem in the geometric setting. We prove that the problem is NP-hard on UDGs and propose constant factor approximation algorithms (refer to Chapter 5).

2.4 Maximum Independent Set Problem

An independent set of a simple undirected graph $G = (V, E)$ is a set of vertices $V' \subseteq V$ such that no two vertices in V' are adjacent in G. The objective of the maximum independent set (MIS) problem is to find an independent set of maximum cardinality in a given graph. A clique is a subset of V such that every pair of vertices in the subset have an edge between them in G. The objective of the maximum clique (MC) problem is to find a clique of maximum cardinality in a given graph. Note that, $V' \subseteq V$ is an independent set in G if and only if V' is a clique in $\overline{G}$ (the complement graph of G).

The MIS and MC problems are NP-complete for general graphs [48]. Garey and Johnson [47] showed that, if there exists a constant factor approximation algorithm for the MC problem, then there exists a polynomial-time approximation scheme (PTAS). Unless NP=Co-R, there can be no polynomial-time algorithm that approximates the MC problem within a factor better than $O(n^{1-\varepsilon})$, for any $\varepsilon > 0$ [53]. Feige et al. [41] showed that, if the MC problem can be approximated within any constant factor in polynomial-time, then $NP = DTIME(n^{O(\log \log n)})$. Arora and Safra [7] showed that approximating the MC problem within any constant factor is NP-hard. Latter, Arora et al. [6] improved the results in [7, 41] and showed that approximating the MC problem within a factor of n^{ε}, where $\varepsilon > 0$, is NP-hard. The best known approximation ratio

till today is $O(\frac{n(\log\log n)^2}{\log^3 n})$ [40]. In summary, all the hardness results of MC are also applicable to MIS in general graphs.

Though the MIS and MC problems have the same limitations in general graphs, they may vary when restricted to special classes of graphs. For instance, the MC problem has polynomial-time algorithms on UDGs, but the MIS problem is NP-hard [27]. Another instance is sparse graphs e.g., planar graphs. The MC problem has a linear time algorithm for planar graphs [25], however, the MIS problem remains NP-hard even for planar graphs of degree at most three [48].

Unlike in the general graphs, constant factor approximation algorithms exist for the MIS problem in UDGs. A simple 5-factor approximation algorithm is proposed in [70] and by taking the advantage of the structure of the given UDG, the authors proposed a heuristic algorithm which has performance guarantee 3. Both the algorithms do not require geometric representation (i.e., coordinates of the disk centers) of the UDG. If the geometric representation is given, the later algorithm runs in $O(n^2)$ time. For a given $(k+1)$-claw free graph ($k \geq 4$) and for every $\varepsilon > 0$, Halldórsson [51] proposed a $(\frac{k}{2}+\varepsilon)$-factor approximation algorithm (that does not require the geometric representation of disks) in time $O(n^{\log_k \frac{1}{\varepsilon}})$ using local improvement search technique for the MIS problem. Therefore, there exists a $(\frac{5}{2}+\varepsilon)$-factor approximation algorithm for UDGs as they are 6-claw free. Most of the work in the literature assumes that the geometric representation of the UDG is given. This assumption allows to partition the plane into grids/strips and solve the problem by solving each grid/strip separately. Matsui [71] considered the MIS problem on UDG defined on a slab (i.e., all the disk centers lie between two parallel lines) of fixed width k, and proposed an algorithm that finds an independent set of maximum cardinality in $O(n^{4^{\lceil \frac{2k}{\sqrt{3}} \rceil}})$ time, where n denotes the number of vertices. For $k < \frac{\sqrt{3}}{2}$, the MIS problem can be reduced to find a longest path problem in directed acyclic graph and hence an MIS can be obtained in $O(n^2)$ time. The author also proposed a $(1-\frac{1}{k})$-factor approximation algorithm for the MIS problem, which runs in $O(kn^{4^{\lceil \frac{2(k-1)}{\sqrt{3}} \rceil}})$ time and uses $O(n^{2k})$ space, for any integer $k \geq 2$. The algorithm can also be extended to

the weighted version of the MIS problem. In the weighted version, each vertex (disk) is associated with a positive weight and the objective is to find an independent set of maximum total weight. We call this problem as the maximum weighted independent set (MWIS) problem. The MIS problem is a special case of the MWIS problem with equal weights.

Das et al. [32] proposed a 2-factor approximation algorithm for the MIS problem with time and space complexities $O(n^3)$ and $O(n^2)$, respectively. Their approach is, (i) split the region into a set of disjoint strips of unit width and compute an MIS for each nonempty strip independently with the aid of dynamic programming, (ii) find the union of the solutions for odd and even strips separately and consider the one with maximum cardinality. The authors also proposed a PTAS with the aid of two-level shifting strategy of Hochbaum and Maass [55]. For any given positive integer $k > 1$, the PTAS produces a solution of size $\frac{1}{(1+\frac{1}{k})^2}|OPT|$ in $O(k^4 n^{\sigma_k \log k} + n \log n)$ time and $O(n + k \log k)$ space, where OPT is an optimum solution and $\sigma_k \leq \frac{7k}{3} + 2$. For the MIS problem on UDG, van Leeuwen [65] proposed a fixed parameter tractable algorithm which runs in $O(t^2 2^{2t} n)$ time, where the parameter t is called the *thickness*[4] of the UDG.

The MIS problem is studied for other intersection graphs too. Cerioli et al. [20] proved that the MIS problem in NP-complete for penny graphs. A *penny graph* is a UDG with the restriction that the disks do not overlap. Das et al. [32] presented a 2-factor approximation result for penny graphs which runs in $O(n \log n)$ time. For a given set R of rectangles of fixed size, Agarwal et al. [1] proposed a 2-factor approximation algorithm for the MIS problem that runs in $O(n \log n)$ time. The authors also proposed a PTAS that computes an independent set of rectangles of size at least $\frac{|OPT|}{(1+\frac{1}{k})}$, for any $k \geq 1$, where $|OPT|$ is the size of a maximum independent set of R. For a given set of arbitrary rectangles of bounded aspect ratio in $\mathbb{R}^d$, Chan [21] proposed a PTAS that runs in $O(n^{\frac{1}{\varepsilon^{d-1}}})$ time and space, where $0 < \varepsilon \ll 1$. Chan and Har-Peled [22] considered the MIS/MWIS problem for pseudo disks in the plane. A set of objects is a collection

[4]A UDG is said to have thickness t, if each strip in the slab decomposition (of width 1) of the UDG contains at most t disk centers.

of *pseudo disks*, if the boundary of every pair of them intersects at most twice. Their algorithm produces a solution of size $(1-\varepsilon)|OPT|$, where $|OPT|$ is the cardinality of an MIS.

Nieberg et al. [77] proposed a PTAS for the MWIS problem on UDG for the case of geometric representation is not given. Erlebach et al. [38] also proposed a PTAS for finding an MWIS in an intersection graph of arbitrary radii disks, based on dynamic programming and the shifting strategy proposed by Hochbaum and Maass [55]. Their approach can be extended to other geometric objects such as squares, regular polygons, and rectangles which are approximately squares.

Chapter 3

Geometric Minimum Dominating Set Problem

In this chapter, we study the geometric version of the minimum dominating set problem and is defined as follows:

> Geometric minimum dominating set (GMDS) problem: *Given a set $\mathcal{P}$ of n points in $\mathbb{R}^2$, find a minimum cardinality subset $\mathcal{P}'$ of $\mathcal{P}$ such that each member of $\mathcal{P}$ lies in the unit disk centered at some member of $\mathcal{P}'$. In other words, the unit disk centered at each member of $\mathcal{P}$ is pierced by at least one member of $\mathcal{P}'$.*

The GMDS problem is a restricted version of the MDS problem in unit disk graph, where an edge $e(p, q)$ implies that the points p and q lie in the intersection region of the disks centered at p and q, respectively. The MDS problem in both graph and geometry has wide range of applications in wireless networks, VLSI, to name a few. Our interest in this problem evolved from the following real life scenario: suppose that a city consists of a set of n buildings and we need to open service centers (say hospitals, fire stations, etc.) in a minimum number of these buildings such that each of the other building is within a predefined distance of at least one service station.

The goal of this chapter is to propose (i) a series of constant factor approximation algorithms, and (ii) a polynomial-time approximation scheme (PTAS) for the GMDS problem.

3.1 Preliminaries

Let $\mathcal{P}$ and $\delta(q)$ denote the given set of n points in $\mathbb{R}^2$ and the unit disk centered at point $q \in \mathcal{P}$, respectively. We say that a point p is covered by q, if $p \in \delta(q)$. For a subset $\mathcal{P}'$ of $\mathcal{P}$ we denote by $\Delta(\mathcal{P}')$, the set of unit disks centered at the points in $\mathcal{P}'$. We say $\mathcal{P}'$ covers $\mathcal{P}$ (or equivalently, $\Delta(\mathcal{P}')$ pierces $\Delta(\mathcal{P})$) if all the points in $\mathcal{P}$ lie within the unit disks centered at the points in $\mathcal{P}'$, i.e., $\mathcal{P} \subseteq \bigcup_{d \in \Delta(\mathcal{P}')} d$. If a subset $\mathcal{P}'$ covers $\mathcal{P}$, then $\mathcal{P}'$ is called as a geometric dominating set (GDS) of $\mathcal{P}$. For any two points p and q in $\mathcal{P}$, let $d(p, q)$ denotes the Euclidean distance between p and q.

3.2 A Simple 5-Factor Approximation Algorithm

In this section, we propose a simple 5-factor approximation algorithm. The worst case and expected case running times of the proposed algorithm are $O(n \log k)$ and $O(n)$, respectively, where k is the size of the output. This a significant improvement over the $O(n \log k)$ time 12-factor approximation result for the GMDS problem given by De et al. [34].

Definition 3.2.1. A GDS $\mathcal{P}'$ is said to be an independent GDS if for each pair $p, q \in \mathcal{P}'$, $p \notin \delta(q)$[1].

Result 3.2.2. For any arbitrary independent GDS $\mathcal{P}'$, $|\mathcal{P}'| \leq 5|OPT|$, where OPT is a GMDS of $\mathcal{P}$.

Proof. The result follows from the fact that for every member $p \in OPT$, there may exist at most 5 disks in an independent GDS of $\Delta(\mathcal{P})$ that can contain the point p. □

[1]Note that, $p \in \delta(q)$ implies $q \in \delta(p)$, and $p \notin \delta(q)$ implies $q \notin \delta(p)$.

We now describe a procedure to compute an independent GDS $\mathcal{P}'$ of $\mathcal{P}$. Let us assume that the points in $\mathcal{P}$ are in the first quadrant, and $\mathcal{R}$ is an axis parallel rectangular region containing $\mathcal{P}$, whose bottom and left boundaries coincide with the x and y-axis of the coordinate system, respectively. We split $\mathcal{R}$ into a grid whose each cell is of size 1×1. The point $p = (p_x, p_y) \in \mathcal{P}$ lies in the grid cell indexed by $[\lfloor p_x \rfloor, \lfloor p_y \rfloor]$. Each non-empty cell is attached with a list of members of $\mathcal{P}$ that are chosen for inclusion in $\mathcal{P}'$, and they lie inside that cell. While considering a point $p \in \mathcal{P}$, we inspect the already chosen members of $\mathcal{P}'$ attached to the cells $[i, j]$, where $\lfloor p_x \rfloor - 1 \leq i \leq \lfloor p_x \rfloor + 1$ and $\lfloor p_y \rfloor - 1 \leq j \leq \lfloor p_y \rfloor + 1$. If there exists at least one disk d in any of those cells that contains the point p, p is not included in $\mathcal{P}'$; otherwise p is included in $\mathcal{P}'$. It is easy to observe that at the end of considering all the members in $\mathcal{P}$, $\mathcal{P}'$ is an independent GDS of $\mathcal{P}$. Note that, (i) a grid cell may contain at most 3 members in $\mathcal{P}'$, and (ii) the number of grid cells to be inspected while processing a point $p \in \mathcal{P}$ is at most 9. We use a height-balanced binary tree to store the indices of the grid cells containing a non-zero number of members in $\mathcal{P}'$. Thus, the time complexity for processing a point $p \in \mathcal{P}$ is $O(\log k)$, where $k = |\mathcal{P}'|$. Thus, we have the following theorem.

Theorem 3.2.3. *The proposed algorithm produces a 5-factor approximation result for the GMDS problem in $O(n \log k)$ time and uses $O(k)$ extra space, where k is the size of the output.*

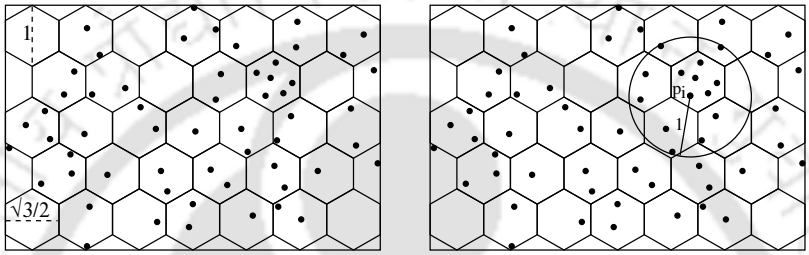
The expected time complexity of the above procedure can be reduced to $O(n)$ by using a hash table of size $O(n)$ for storing the grid cells containing non-zero members of $\mathcal{P}'$ instead of a height-balanced binary tree.

3.3 A 4-Factor Approximation Algorithm

In this section, we present a 4-factor approximation algorithm for the GMDS problem. The algorithm is an in-place algorithm and the worst case running time is $O(n^6 \log n)$.

We partition the region $\mathcal{R}$ containing the point set $\mathcal{P}$ into regular hexagonal cells of side length $\frac{1}{2}$ such that no point of $\mathcal{P}$ lies on the boundary of any cell (see Figure 3.1 (a)). The following observation is trivial as the length of a longest diagonal in any cell is of unit length.

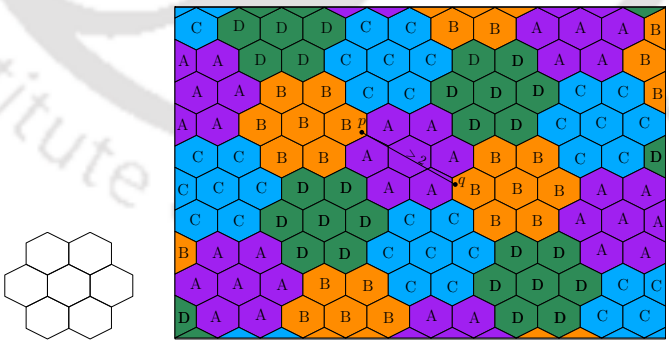
Observation 3.3.1. All the points inside a cell can be covered by a unit radius disk centered at any point inside that cell (see Figure 3.1 (b)).



(a) (b)

Figure 3.1: (a) Hexagonal partition of the rectangular region containing the points, and (b) a unit disk centered at a point p_i circumscribes the cell in which p_i lies.

Definition 3.3.2. A *septa-hexagon* is a combination of 7 adjacent cells such that one cell is surrounded by six other cells (see Figure 3.2 (a)).



(a) (b)

Figure 3.2: (a) A septa-hexagon, and (b) a septa-hexagonal partition of $\mathcal{R}$ and its 4-coloring scheme.

Let us consider a septa-hexagonal partitioning of $\mathcal{R}$ such that no point of $\mathcal{P}$ lies on the boundary of any septa-hexagon, and a 4 coloring scheme of it (see Figure 3.2 (b)). Consider a septa-hexagon $\mathcal{H}$ which is assigned the color A. Its adjacent six septa-hexagons are assigned three colors, namely B, C, and D, such that pair of opposite septa-hexagons adjacent to $\mathcal{H}$ are assigned the same color. Let $\mathcal{H}'$ and $\mathcal{H}''$ be two septa-hexagons having the same color (say B) adjacent to $\mathcal{H}$. Since no point of $\mathcal{P}$ lies on the boundary of $\mathcal{H}'$ and $\mathcal{H}''$, the minimum distance between two points $p \in \mathcal{H}' \cap \mathcal{P}$ and $q \in \mathcal{H}'' \cap \mathcal{P}$ is greater than 2 (see Figure 3.2 (b)). This leads to the following observation.

Observation 3.3.3. If $\mathcal{H}'$ and $\mathcal{H}''$ are two septa-hexagons having the same color, then a single unit disk cannot cover points in $\mathcal{H}'$ and $\mathcal{H}''$ simultaneously.

The basic idea of the proposed algorithm is, consider each septa-hexagon $\mathcal{H}$, and compute a subset of points in $\mathcal{P}$ of minimum size that covers all the points in $\mathcal{P} \cap \mathcal{H}$. We refer this problem as a *single septa-hexagon* GMDS problem. Finally, we report the set $\mathcal{T}$ which is the union of the solutions for all septa-hexagons.

Observation 3.3.3 says that if OPT_A denotes a minimum size subset of points in $\mathcal{P}$ such that OPT_A covers all the points of $\mathcal{P}$ lying in the septa-hexagons colored with A, then OPT_A is the union of the optimum solutions of the GMDS problems for all the septa-hexagons colored A. The same result holds for the other three colors also.

Lemma 3.3.4. *$|\mathcal{T}| \leq 4|OPT|$, where OPT is a subset of $\mathcal{P}$ of minimum size such that OPT covers $\mathcal{P}$.*

Proof. The result follows from the facts that (i) $\mathcal{T} = OPT_A \cup OPT_B \cup OPT_C \cup OPT_D$, and (ii) $|OPT_i| \leq |OPT|$ for all $i = A, B, C, D$. □

3.3.1 Computing an optimum solution for a single septa-hexagon GMDS problem

Consider a septa-hexagon $\mathcal{H}$. Let S_1 and S_2 be two subsets of $\mathcal{P}$ such that S_1 contains the points lying in $\mathcal{H}$ and S_2 contains all the points that are coverable from the points

in S_1. That is, $S_1 = \{p \mid p \in \mathcal{P} \cap \mathcal{H}\}$ and $S_2 = \{p \mid p \in \mathcal{P}$ and $\delta(p) \cap S_1 \neq \emptyset\}$. Surely, $S_1 \subseteq S_2$. The following lemma is due to Observation 3.3.1.

Lemma 3.3.5. *If $OPT_{\mathcal{H}}$ is a subset of S_2 of minimum size such that $OPT_{\mathcal{H}}$ covers S_1, then $|OPT_{\mathcal{H}}| \leq 7$.*

Given an array containing the points in $\mathcal{P}$, in $O(n)$ time we can arrange these points such that the points in S_1 are placed at the beginning followed by the points in $S_2 \setminus S_1$.

As suggested in Lemma 3.3.5, we consider all possible combinations of points in S_2 of size $i = 1, 2, \ldots, 6$, respectively in this order. For each combination, we check whether all the points in S_1 are covered or not. While considering the number i, if there exists a subset of S_2 of size i that covers all the points in S_1, then that subset is reported and execution stops. If this fails for all $i = 1, 2, \ldots, 6$, then the optimum solution of the septa-hexagon $\mathcal{H}$ consists of any one point from each non-empty cell of $\mathcal{H}$. This trivial algorithm needs $O(n^7)$ time.

We can reduce the time complexity as follows: let the points of S_2 are stored at the beginning of the array containing the points in $\mathcal{P}$ such that the points in each cell of $\mathcal{H}$ are stored consecutively. As earlier, we generate different combinations of points of size $i = 1, 2, \ldots, 5$ from S_2 in this order. If any one of these combinations can produce a solution, the process terminates. Otherwise, we again consider different combinations of size 5. For each combination, say X, the uncovered points of S_1 in each cell of $\mathcal{H}$ are moved at the beginning of the array, and the convex hull of the uncovered points is computed in an in-place manner in $O(n \log n)$ time [15]. Now, each point p in $S_2 \setminus X$ is chosen, and the farthest uncovered point q in S_1 is identified. Note that q is a vertex of the convex-hull computed. If $d(p, q) \leq 1$, return the solution X together with q and terminate. This needs $O(\log n)$ time for each point in $S_2 \setminus X$. If the covering is not done yet, we chose one point from each non-empty cell of $\mathcal{H}$.

Lemma 3.3.6. *For a given set $\mathcal{P}$ of n points and a septa-hexagon $\mathcal{H}$, a GMDS for $\mathcal{P} \cap \mathcal{H}$ can be computed using the points in S_2 in $O(m^6 \log m)$ time using $O(1)$ extra space, where $m = |S_2|$.*

Theorem 3.3.7. *The proposed 4-coloring scheme gives a 4-factor approximation algorithm for the GMDS problem in $O(n^6 \log n)$ time using $O(1)$ extra space, where n is the size of the input.*

Proof. The approximation factor follows from Lemma 3.3.4. The time complexity result of the theorem follows from Lemma 3.3.6 and the fact that each point in $\mathcal{P}$ can participate in the computation for at most a constant number of septa-hexagons. □

3.3.2 A $\frac{14}{3}$-factor approximation algorithm

We now show that a slight modification of the earlier algorithm produces a $\frac{14}{3}$-factor approximation result for the GMDS problem in $O(n^5 \log n)$ time. First we find a feasible solution for a single septa-hexagon, then we use the union of these solutions to provide a feasible solution for the GMDS problem. We prove that the bound on the ratio of this feasible solution to an optimum solution is $\frac{14}{3}$.

For a single septa-hexagon $\mathcal{H}$, we try to find a solution of size at most 5, if exists. This can be done by considering all the possible combinations of size $i = 1, 2, 3, 4$, respectively in this order from S_2 as in the previous section. For each combination X, the uncovered points in S_1 are examined to be covered by a single point in $S_2 \setminus X$. If there is a point $q \in S_2 \setminus X$ which covers all the uncovered points in S_1, then X together with q is reported. After considering all possible combinations of 4 points in S_2, if the covering is not done, then we arbitrarily choose one point from each non-empty cell of $\mathcal{H}$ and report them. Thus, we have the following lemma.

Lemma 3.3.8. *For a given set $\mathcal{P}$ of n points and a septa-hexagon $\mathcal{H}$, a set $S_{\mathcal{H}} \subseteq S_2$ of size at most 5 points such that $S_{\mathcal{H}}$ covers all the points in $\mathcal{P} \cap \mathcal{H}$, can be tested in $O(m^5 \log m)$ time using $O(1)$ extra space, where $m = |S_2|$.*

Theorem 3.3.9. *The proposed 4-coloring scheme gives a $\frac{14}{3}$-factor approximation algorithm for the GMDS problem in $O(n^5 \log n)$ time using $O(1)$ extra space, where n is the input size.*

Proof. If the above algorithm cannot obtain a solution of size at most five for a septa-hexagon $\mathcal{H}$, then $|OPT_{\mathcal{H}}| \geq 6$. If $|OPT_{\mathcal{H}}| \geq 6$, our algorithm returns a solution, say $SOL_{\mathcal{H}}$, of size 7. Thus, for any sept-hexagon $\mathcal{H}$, if $|OPT_{\mathcal{H}}| \geq 6$ implies $|SOL_{\mathcal{H}}| = 7$, and hence the approximation factor is $\frac{|SOL_{\mathcal{H}}|}{|OPT_{\mathcal{H}}|} \leq \frac{7}{6}$. Let SOL_A, SOL_B, SOL_C, and SOL_D be the union of the solutions of all the septa-hexagons colored A, B, C, and D, respectively. Since $|SOL_A| = \sum |SOL_{\mathcal{H}}|$, where the sum is taken over all the septa-hexagons colored A, implies, $|SOL_A| \leq \frac{7}{6}|OPT|$. Similarly, the bound also holds for the remaining three colors B, C, and D. Thus the theorem follows. □

3.4 A 3-Factor Approximation Algorithm

In this section, we show that using the similar technique as in Section 3.3, the running time of getting a 3-factor approximation result can be improved to $O(n^{11} \log n)$ over the existing $O(n^{15} \log n)$ time algorithm.

Definition 3.4.1. A *super-cell* is a combination of 15 regular hexagons of side length $\frac{1}{2}$ arranged in three consecutive rows as shown in Figure 3.3 (a).

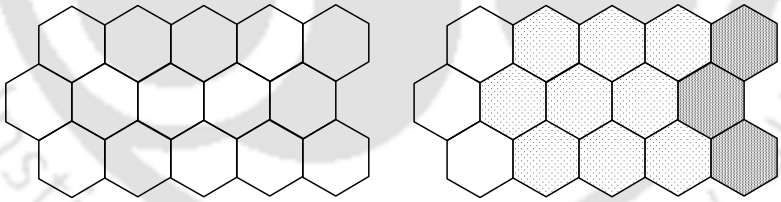


Figure 3.3: (a) An example of a super-cell, and (b) decomposition of a super-cell.

Consider a *super-cell* $\mathcal{D}$. Let S_1 and S_2 be two subsets of $\mathcal{P}$ such that S_1 contains the points lying in $\mathcal{D}$ and S_2 contains all the points that are coverable from the points in S_1. That is, $S_1 = \{p \mid p \in \mathcal{P} \cap \mathcal{D}\}$ and $S_2 = \{p \mid p \in \mathcal{P}$ and $\delta(p) \cap S_1 \neq \emptyset\}$. Note that, $S_1 \subseteq S_2$.

We compute a subset $OPT_{\mathcal{D}}(\subseteq S_2)$ of minimum size such that $OPT_{\mathcal{D}}$ covers S_1. We refer this problem as a *single super-cell GMDS* problem. We show that the union of the optimum solutions of all the single super-cell GMDS problems gives a 3-factor approximation result.

Lemma 3.4.2. *If $OPT_{\mathcal{D}}$ is a subset of S_2 of minimum size such that $OPT_{\mathcal{D}}$ covers S_1, then $|OPT_{\mathcal{D}}| \leq 15$.*

Proof. The lemma follows from Observation 3.3.1 and the fact that the super-cell $\mathcal{D}$ has at most 15 non-empty cells. □

Let us consider a super-cell partition of $\mathcal{R}$ such that no point of $\mathcal{P}$ lies on the boundary of any super-cell, and a 3-coloring scheme of it using colors A, B, and C (see Figure 3.4). Consider a super-cell colored A. Its adjacent super-cells are assigned colors B, and C alternately.

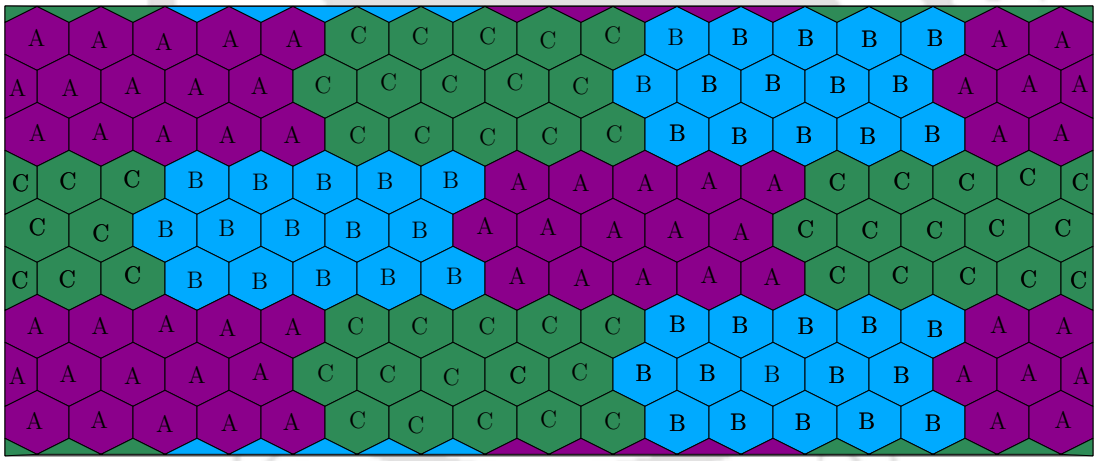


Figure 3.4: A super-cell partition and its 3-coloring scheme.

Observation 3.4.3. If $\mathcal{D}'$ and $\mathcal{D}''$ are two super-cells having the same color, then a single unit disk cannot cover points in $\mathcal{D}'$ and $\mathcal{D}''$ simultaneously.

3.4.1 Computing an optimum solution for a single super-cell GMDS problem

We decompose a super-cell $\mathcal{D}$ into 3 regions namely $\mathcal{D}^1, \mathcal{D}^2$, and $\mathcal{D}^3$ as shown in Figure 3.3 (b), these regions are demonstrated using unshaded, light-shaded, and dark-shaded regions, respectively.

Lemma 3.4.4. *For any unit radius disk d and a super-cell $\mathcal{D}$, $(\mathcal{D}^1 \cap \mathcal{D}^3) \cap d = \emptyset$, i.e., there does not exist any unit disk d that has intersection with both $\mathcal{D}^1$ and $\mathcal{D}^3$.*

Proof. The lemma follows from the fact that if p and q are two arbitrary points in $\mathcal{D}^1$ and $\mathcal{D}^3$, respectively, then $d(p, q)$ is greater than 2. □

Let $S_1^1 = S_1 \cap \mathcal{D}^1$, $S_1^2 = S_1 \cap \mathcal{D}^2$, and $S_1^3 = S_1 \cap \mathcal{D}^3$. A point on a boundary can be assigned to any one of the sets associated with that boundary. Let $S_2^1 = \{p \mid p \in S_2$ and $\delta(p) \cap S_1^1 \neq \emptyset\}$, $S_2^2 = \{p \mid p \in S_2$ and $\delta(p) \cap S_1^2 \neq \emptyset\}$, and $S_2^3 = \{p \mid p \in S_2$ and $\delta(p) \cap S_1^3 \neq \emptyset\}$. By lemma 3.4.4, $S_2^1 \cap S_2^3 = \emptyset$.

We first compute the sets $S_1^1, S_1^2, S_1^3, S_2^1, S_2^2$, and S_2^3 in $O(n)$ time as in the case of sept-hexagon in Section 3.3. These sets can be arranged in the input array in the aforesaid order. Next, we consider each possible combination $X = \{p_1, p_2, \ldots, p_j\}$ of j points in S_2^2, where $1 \leq j \leq 9$. Now,

- If $\Delta(S_1)$ is pierced by $\Delta(X)$, then X is reported.
- If $\Delta(S_1^2)$ is not pierced by $\Delta(X)$, then X is ignored.
- If $\Delta(S_1^2)$ is pierced by $\Delta(X)$ but $\Delta(S_1)$ is not pierced by $\Delta(X)$, then we need to choose more disks to pierce the remaining disks as follows.

Let $\hat{S_1^1} \subseteq S_1^1$ and $\hat{S_1^3} \subseteq S_1^3$ be such that the members in $\Delta(\hat{S_1^1})$ and $\Delta(\hat{S_1^3})$ are not pierced by $\Delta(X)$. We need to pierce them by using (the unit disks centered at) the points in S_2^1 and S_2^3, respectively. As in the algorithm of Section 3.3, we can find the minimum number of points of S_2^1 (resp. S_2^3) for piercing $\Delta(\hat{S_1^1})$ (resp. $\Delta(\hat{S_1^3})$) in $O(m^2 \log m)$

time, where $m = |S_2^1|$ (resp. $m = |S_2^3|$). Finally, we report the union of the solutions of minimum cardinality.

Lemma 3.4.5. *For a given set $\mathcal{P}$ of n points and a super-cell $\mathcal{D}$, a GMDS for $\mathcal{P} \cap \mathcal{D}$ can be computed using the points in S_2 in $O(m^{11} \log m)$ time using $O(1)$ extra space, where $m = |S_2|$.*

Theorem 3.4.6. *The 3-coloring scheme gives a 3-factor approximation algorithm for the GMDS problem in $O(n^{11} \log n)$ time using extra $O(1)$ space, where n is the input size.*

3.4.2 A $\frac{45}{13}$-factor approximation algorithm

As in Subsection 3.3.2, we can reduce the running time of the GMDS problem by a factor of n by scarifying the approximation factor a bit. As in the earlier algorithm, while computing a solution for a super-cell, we consider each possible combination $X = \{p_1, p_2, \ldots, p_j\}$ of j points in S_2^2, where $1 \leq j \leq 9$. For a particular combination X, if $\Delta(X)$ pierces $\Delta(S_1^2)$ but not $\Delta(S_1)$, then we need to choose more disks to pierce the remaining disks. If $\hat{S_1^1} \subseteq S_1^1$ and $\hat{S_1^3} \subseteq S_1^3$ are not covered by X, we compute the convex hulls of the points in $\hat{S_1^1}$ and $\hat{S_1^3}$. Now, we consider each point $p \in S_2^1$ (resp. $p' \in S_2^3$), and check whether the distance of p from its furthest point $q \in \hat{S_1^1}$ (resp. $q' \in \hat{S_1^3}$) is less than or equal to 1 or not. If so, report the solution as $X \cup \{p, p'\}$; otherwise report X along with one point from each cell of $\mathcal{D}_1$, and one point from each cell of $\mathcal{D}_3$. Thus, if the size of the optimum solution is greater that or equal to $|X| + 4$, we are choosing a solution of size at most $|X| + 6$, where $|X| \leq 9$. This leads to an approximation ratio $\frac{|X|+6}{|X|+4} = \frac{15}{13}$, when $|X| = 9$.

Theorem 3.4.7. *The proposed 3-coloring scheme gives a $\frac{45}{13}$-factor approximation algorithm for the GMDS problem in $O(n^{10} \log n)$ time using $O(1)$ extra space, where n is the input size.*

3.5 Shifting Strategy and its Application to the GMDS Problem

The shifting strategy, a divide-and-conquer paradigm, was introduced by Hochbaum and Maass [55] and used it to obtain a PTAS for many covering and packing problems on the low-dimensional Euclidean space. We first discuss the shifting strategy in the context of the GMDS problem. Next, using a two-level shifting strategy (which is a generalization of the shifting strategy), we deduce a $\frac{5}{2}$-factor approximation result and a PTAS for the GMDS problem.

3.5.1 The shifting strategy

The basic idea behind the shifting strategy is, decompose the original problem into several sub-problems that are solvable or at least a good approximated solution is possible in polynomial-time. The union of the solutions of the sub-problems is a feasible solution to the original problem with respect to the decomposition considered. For a given $\varepsilon > 0$, the strategy considers all distinct possible decompositions and returns the best feasible solution among all the decompositions considered.

Definition 3.5.1. A *monotone chain* c with respect to a line L is a chain of line segments such that any line perpendicular to L intersects it only once. We define the *distance* d between a pair of monotone chains c' and c'' as the minimum Euclidean distance between any two points p' and p'' on the chains c' and c'', respectively. A *monotone strip* M is the region bounded by a pair of monotone chains c' and c'' such that the area is left closed and right open. The *width* of a strip M is the distance between c' and c''.

Let $c_1, c_2, \ldots, c_r$ be r consecutive monotone chains with respect to the y-axis in left to right order that splits the region $\mathcal{R}$ such that c_1 and c_r are the left and right boundary of $\mathcal{R}$, respectively (see Figure 3.5), and the distance d between each pair of consecutive monotone chains is at least 2. Now, we have the following result.

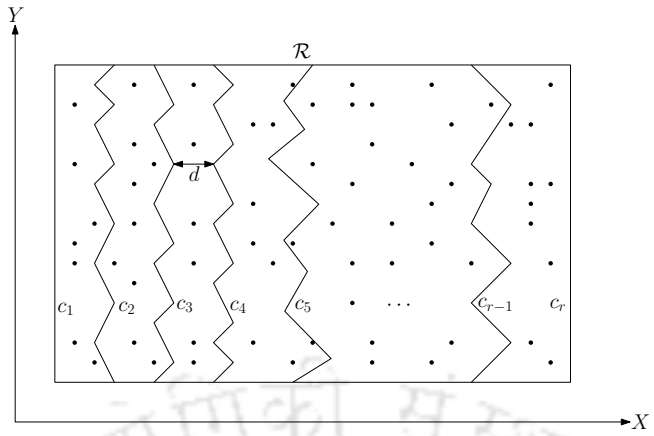


Figure 3.5: Demonstration of shifting strategy.

Theorem 3.5.2. *For a given point set $\mathcal{P}$ in $\mathbb{R}^2$, if there exists an algorithm $\mathcal{A}$ that can produce an α-factor approximation result for the GMDS problem defined for the points inside a monotone strip of* width *ℓd, where each pair of consecutive monotone chains are at distance $d \geq 2$ and ℓ is an integer (shifting parameter), then there exists an $\alpha(1 + \frac{1}{\ell})$-factor approximation algorithm for the GMDS problem for the point set $\mathcal{P}$.*

Proof. The algorithm is similar to the algorithm proposed by Hochbaum and Maass [55]. Since the monotone strips are of width ℓd and the distance between a pair of monotone strips is $d \geq 2$, the solution of one strip does not affect the solution of any other strip. Thus the approximation factor of the algorithm follows. □

3.5.2 A $\frac{5}{2}$-factor approximation algorithm

Here we propose a $\frac{5}{2}$-factor approximation algorithm for the GMDS problem for a given set $\mathcal{P}$ of n points in $\mathbb{R}^2$ using shifting strategy discussed in Subsection 3.5.1.

Definition 3.5.3. A *duper-cell $\mathcal{E}$* is a combination of 30 cells (regular hexagons of side length $\frac{1}{2}$) as shown in Figure 3.6.

The basic idea of the proposed $\frac{5}{2}$-factor approximation algorithm is as follows: first optimally solve the sub-problem for a *duper-cell $\mathcal{E}$*, we refer this problem as a *single*

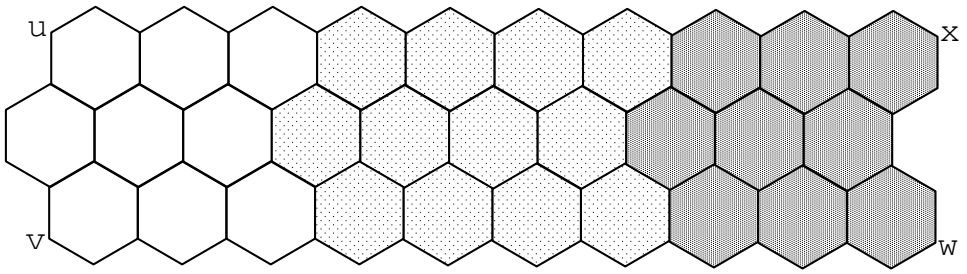


Figure 3.6: Demonstration of a duper cell and its partition.

duper-cell GMDS problem; then apply shifting strategy in both horizontal and vertical directions separately for the entire region $\mathcal{R}$, and use the optimal algorithm for solving the generated sub-problems consisting of duper-cells.

3.5.2.1 Computing an optimum solution for a single duper-cell GMDS problem

We divide the duper-cell $\mathcal{E}$ into three regions as in the case of super-cell in Section 3.4. Let us name these regions unshaded, light-shaded, and dark-shaded regions, respectively (see Figure 3.6). The algorithm for computing an optimum solution for a single duper-cell is similar to the algorithm for a single super-cell (refer Section 3.4). Thus, we have the following lemma.

Lemma 3.5.4. *A GMDS for the set of points inside a duper-cell $\mathcal{E}$ can be computed optimally in $O(m^{20} \log m)$ time and using $O(1)$ extra space, where m is the number of points inside the duper-cell $\mathcal{E}$.*

Now, we discuss the method of applying two-level shifting strategy to the GMDS problem. Split the entire region $\mathcal{R}$ into strips using x-monotone chains, say $c_1, c_2, \ldots, c_r$, such that (i) each segment is of length $\frac{1}{2}$ unit, (ii) each pair of consecutive segments make an angle $\frac{2\pi}{3}$, (iii) the distance between any two consecutive chains is $\frac{1}{2}$ unit, and (iv) every chain c_{2i} starts at exactly one unit away from c_{2i-1} in the opposite direction (either up or down) of c_{2i-1}. Next, split each strip into hexagonal regions of side length $\frac{1}{2}$.

In the first level of the shifting strategy, we take 3 consecutive rows of hexagons as a single horizontal strip and in the second level we take 10 columns of hexagons as a vertical strip inside that horizontal strip, i.e., we partition the horizontal strip into duper-cells. Each dupe-cell in the partition is bounded by four monotone chains, uv, vw, wx and xu, where uv and wx are monotone with respect to x-axis, and vw and xu are monotone with respect to y-axis (see Figure 3.6). We name these chains as **left, bottom, right**, and **top**, respectively. Thus, the region $\mathcal{R}$ is split into duper-cells. By applying Theorem 3.5.2 in two-levels, we have the following result.

Theorem 3.5.5. *A $\frac{5}{2}$-factor approximation result for the GMDS problem can be obtained in $O(n^{20}\log n)$ time, where n is the input size.*

Proof. In our GMDS problem the disks are of diameter 2. Let ℓ_1 and ℓ_2 be two shifting parameters in the first and second level shifting strategies, respectively. In the first level of partitioning we consider 3 consecutive rows of hexagons as a single horizontal strip, hence the distance between the monotone chains bottom and top bounding the strip is 2. Implies, $\ell_1 = 1$ as $d = 2$ and $\ell_1 d = 2$. Again, inside a monotone strip of the first level, the distance between the monotone chains left and right corresponding to a duper-cell is greater than 8 i.e., $\ell_2 d > 8$, implies, $\ell_2 > 4$. At the end of the second level, the region $\mathcal{R}$ is partitioned into duper-cells. We have an algorithm to compute an optimum solution for a duper-cell. Thus, the application of Theorem 3.5.2 in both the levels yields the approximation factor and is less than $1(1+\frac{1}{1})(1+\frac{1}{4}) = \frac{5}{2}$. The time complexity $O(n^{20}\log n)$ follows from Lemma 3.5.4. □

3.5.3 A PTAS for the GMDS problem

We apply two-level shifting strategy (as in the previous subsection) to obtain a PTAS for the GMDS problem by solving the GMDS problem optimally for the points inside a region (say $\mathcal{F}$) bounded by two pairs of monotone chains such that the distance between **left** and **right** (resp. **bottom** and **top**) monotone chains is k (see Figure 3.7), where k

is an integer given as input. In the first level, we partition $\mathcal{R}$ into horizontal strips of width k using x monotone chains. In the second level, we further divide each horizontal strip into regions of size $k \times k$ using y monotone chains. We solve each sub-problem independently and get an optimum solution; this yields a $(1+\frac{1}{k})^2$-factor approximation algorithm for the GMDS problem for the input $\mathcal{P}$.

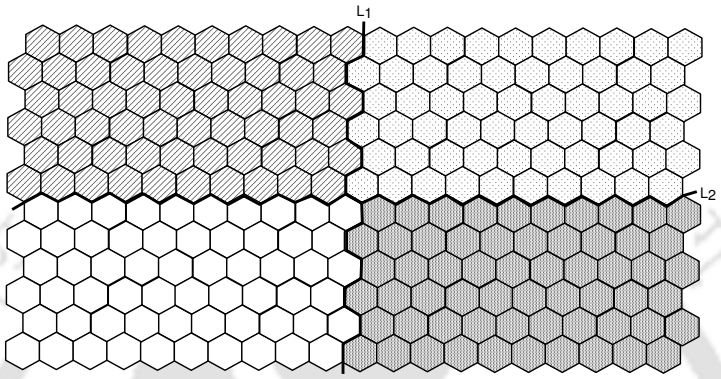


Figure 3.7: Demonstration of the PTAS.

3.5.3.1 Computing an optimum solution for $\mathcal{P} \cap \mathcal{F}$

In order to solve the GMDS problem for the points in $\mathcal{P} \cap \mathcal{F}$ optimally, we further decompose $\mathcal{F}$ into four quadrants using the monotone chains L_1 and L_2 as shown in Figure 3.7. The number of unit disks in an optimum solution that pierce all the unit disks which intersecting the chain L_1 with centers **left** (resp. **right**) side of L_1 is at most $\lceil 2 \times 2 \times \frac{2k}{\sqrt{3}} \rceil$, which is less than $5k$, and the number of disks in an optimum solution that pierce all the unit disks which intersecting the chain L_2 with centers **bottom** (resp. **top**) side of L_2 is at most $\lceil 2 \times 2 \times \frac{2k}{\sqrt{3}} \rceil$, which is less than $5k$. That is, in any optimum solution the set, say $S(\subseteq \mathcal{P} \cap \mathcal{F})$, of points which cover all the points that are of distance at most 1 from the chains L_1 and L_2 is at most $10k$. Note that $|S|$ is independent of the number of points in $\mathcal{P} \cap \mathcal{F}$. We consider all possible combinations of points in S of size at most $10k$, and for every combination we do the following: consider the combination as part of the solution and delete the points in the quadrants that are within unit distance

from the chosen points. Compute an optimum solution for the rest of the points in the quadrants separately using the same procedure (note that any unit disk centered in the optimum solution of any quadrant does not intersect L_1 and L_2, and hence we can solve them independently). Next, we check whether the solutions obtained in the quadrants together with the chosen combination covers the points in $\mathcal{P} \cap \mathcal{F}$. We consider a best possible solution once the process ends. If $T(n,k)$ is the running time of the recursive algorithm for the GMDS problem for $\mathcal{P} \cap \mathcal{F}$, then we have the following recurrence relation: $T(n,k) = 4 \times T(n, \frac{k}{2}) \times n^{10k}$. This leads to the following theorem.

Theorem 3.5.6. *For a given set $\mathcal{P}$ of n points in $\mathbb{R}^2$, the proposed algorithm produces a solution in $n^{O(k)}$ time, whose size is at most $(1+\frac{1}{k})^2 \times |OPT|$, where k is a positive integer depends on ε and OPT is an optimum solution.*

Though the running time of the proposed PTAS is same as the running time of the PTAS proposed by De et al. [34] in terms of O notation, but the constant involved in O is smaller than [34].

3.6 Conclusion

In this thesis, we proposed a series of constant factor approximation algorithms for the GMDS problem. We first presented a simple $O(n \log k)$ time 5-factor approximation algorithm for this problem, where k is the size of the output. Next, we presented a simple 4-factor and 3-factor approximation algorithms in $O(n^6 \log n)$ and $O(n^{11} \log n)$ time, respectively, which improved the time complexities of best known result by a factor of $O(n^2)$ and $O(n^4)$, respectively [34]. We have also presented a $\frac{14}{3}$-factor and $\frac{45}{13}$-factor approximation algorithms in time $O(n^5 \log n)$ and $O(n^{10} \log n)$, respectively. Finally, we proposed a two-level shifting lemma and using this lemma we presented a $\frac{5}{2}$-factor approximation algorithm and a PTAS for the GMDS problem.

Chapter 4

Distributed Minimum Connected Dominating Set Problem

In this chapter, we study distributed construction of a connected dominating set (CDS) in a given unit disk graph (UDG). Unit disk graph is a mathematical model for a homogeneous wireless ad-hoc network (WANET) [27]. That is, the network topology of a WANET can be modeled as a UDG, $G = (V, E)$, where each node represents a wireless device and there is an edge between two nodes if and only if one node is in the communication range of the other. In network terminology, two devices in a wireless ad-hoc networks communicate if they are in each others transmission range. As WANETs do not have any physical backbone infrastructure, unlike in wired networks and thus, a CDS can act as a virtual backbone in WANETs. Since the nodes in WANETs are equipped with limited battery power, constructing a CDS of minimum size is vital for longer operational lifetime. Accordingly, we define the minimum connected dominating set (MCDS) problem formally as follows:

> Given a connected UDG, $G = (V, E)$, find a minimum cardinality set $D \subseteq V$ satisfying (i) D is a dominating set of G, and (ii) the induced subgraph of D is connected.

This chapter targets to design a distributed algorithm which constructs a CDS in $O(\Delta)$ time and has $O(n)$ message complexity.

4.1 Network Model

We assume that all the nodes (i.e., devices) are deployed in a rectangular region in the plane and they have an equal transmission range of one unit. We also assume (i) each node has unique ID and an absolute location information in the plane, (ii) all the nodes in the network are stationary, (iii) the communication between two nodes is achieved through a single radio transmission if the nodes are close enough to receive each other's transmission, and (iv) a node knows the IDs and location information of its 1-hop neighbors and the corners of the rectangular region. The model of distributed computation that we adopt is the standard fully synchronous model and we do not consider collisions or other transmission failures in the network. The aforesaid assumptions in our network model are feasible and are widely used in the literature [62].

4.2 Definitions and Preliminaries

Let $\mathcal{P}$ be a set of n points (nodes) in $\mathbb{R}^2$. Let $x(p)$ and $y(p)$ denote the x and y coordinates of the point $p \in \mathcal{P}$. For any two points p and q in $\mathcal{P}$ we denote by $d(p, q)$, the Euclidean distance between p and q. Let $G = (V, E)$ be the UDG corresponding to the given set $\mathcal{P}$, where $V = \mathcal{P}$, and there is an edge between two points if the Euclidean distance between them is less than or equal to 1 (see Figure 4.1(a) and Figure 4.1(b)). We use points and nodes interchangeably in the context of UDG and WANETs, in the rest of the chapter. Let $\mathcal{R}$ be the rectangular region containing the point set $\mathcal{P}$. Consider a partitioning of the rectangular region $\mathcal{R}$ into regular hexagons of side length $\frac{1}{2}$ as shown in Figure 4.1(c). We refer to each regular hexagon in the hexagonal tiling as a *cell*, namely H_i, where i is its label. Since a node knows its coordinates, it can determine the cell in which it lies.

Observation 4.2.1. All nodes in a cell are within the communication (transmission) range of any other node in that cell (see Figure 4.1(d)).

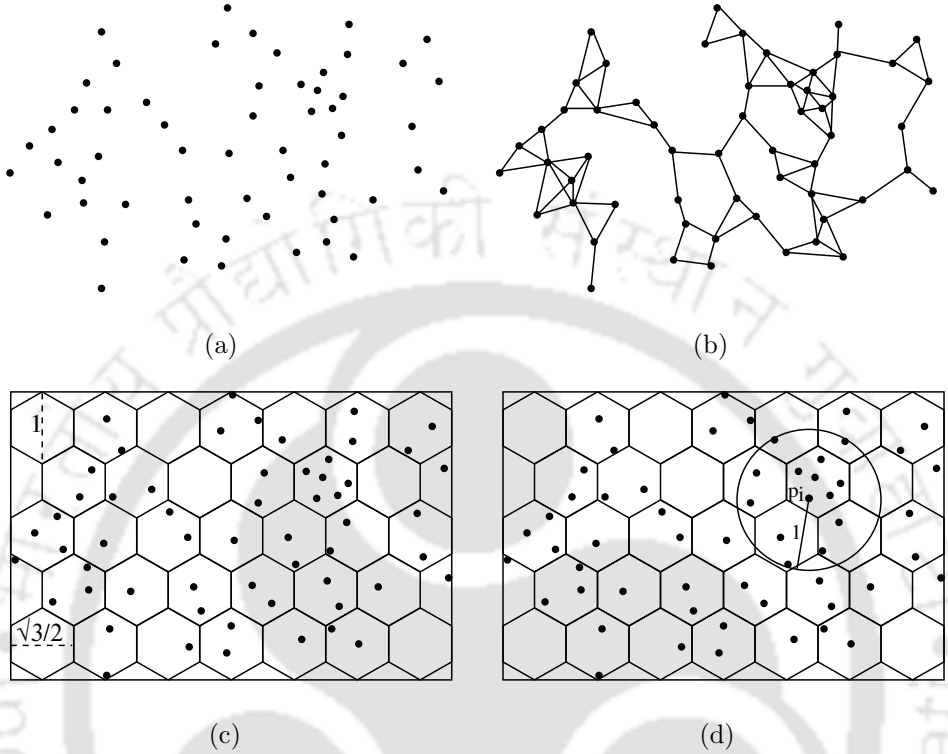


Figure 4.1: (a) A set of points (nodes) in the plane, (b) the UDG corresponding to the points, (c) hexagonal partition of the rectangular region containing the points, and (d) a unit disk centered at a point p_i circumscribes the cell in which p_i lies.

The proposed distributed algorithm to obtain a CDS is divided into two phases: (i) construction of a dominating set (DS): one node is selected from every non-empty cell H_i to cover the rest of the nodes in that cell. We refer to this node as the dominator of that cell and is denoted by d_i, where $d_i \in H_i$. Therefore, the set of all such dominators is a dominating set $\mathcal{D}$ of G, and (ii) connect the nodes obtained in (i) by a subset of nodes in $V \setminus \mathcal{D}$. In the rest of the chapter, we refer to the DS obtained in (i) as $\mathcal{D}$, and the connector nodes chosen in (ii) as $\mathcal{C}$. The dominating set $\mathcal{D}$, together with the connector set $\mathcal{C}$ forms a CDS, say $\mathcal{M}$. Therefore, $\mathcal{M} = \mathcal{D} \cup \mathcal{C}$ and $|\mathcal{M}| = |\mathcal{D}| + |\mathcal{C}|$.

The following definitions, Lemma 4.2.11 and Lemma 4.2.12 play crucial role to find the set $\mathcal{C}$ of connector nodes during the second phase.

Definition 4.2.2. A pair of dominators (d_i, d_j) such that $d_i \in H_i$ and $d_j \in H_j$ is said to be a pair of *k-hop dominators* if the number of edges on a shortest path between d_i and d_j in G is at most k.

Definition 4.2.3. A pair of 2-hop dominators (d_i, d_j) such that $d_i \in H_i$ and $d_j \in H_j$ is said to be a pair of *2-hop special dominators* if the node connecting them, say u, is either in H_i or H_j. The path $d_i \sim u \sim d_j$ is said to be a 2-hop special path (see Figure 4.2(a)).

Definition 4.2.4. A pair of 3-hop dominators (d_i, d_j) such that $d_i \in H_i$ and $d_j \in H_j$ is said to be a pair of *3-hop special dominators* if there exist two nodes u and v such that $u \in H_i$, $v \in H_j$, and u, v are adjacent. The path $d_i \sim u \sim v \sim d_j$ is said to be a 3-hop special path (see Figure 4.2(b)) and the nodes u and v are called connector nodes.

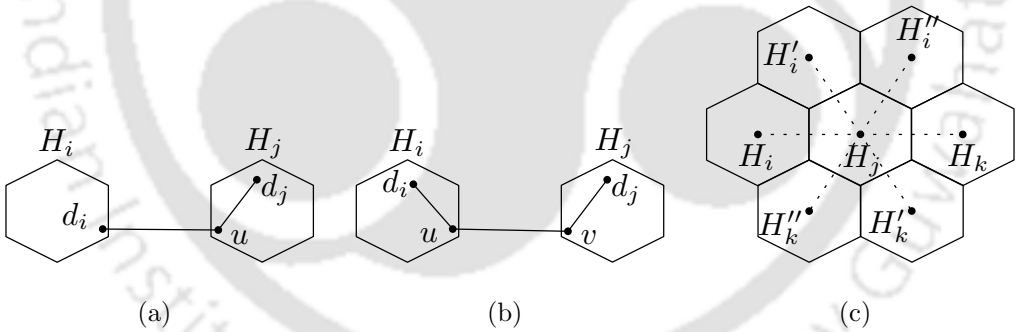


Figure 4.2: (a) d_i, d_j are a pair of 2-hop special dominators and u is a connector node, and (b) d_i, d_j are a pair of 3-hop special dominators and u, v are connector nodes.

We use **(1, 2, 3) - special dominators** to represent 1-hop, 2-hop special, and 3-hop special dominators, and $SD(d_i)$ to represent the set of all (1, 2, 3) - special dominators of the dominator d_i. In Figure 4.3, the set of all special dominators of d_i are $d_j, d_k, d_l, d_m, d_n, d_o$, and d_p, hence, $SD(d_i) = \{d_j, d_k, d_l, d_m, d_n, d_o, d_p\}$. Note that d_i and

d_q are 2-hop dominators but not 2-hop special dominators. Similarly, d_i and d_r are not 3-hop special dominators as the node adjacent to d_r does not lie in the cell where d_r lies.

Definition 4.2.5. A special path between two nodes in $SD(d_i)$ is a path having the connector nodes in the same cells as that of the dominators (see Figure 4.3).

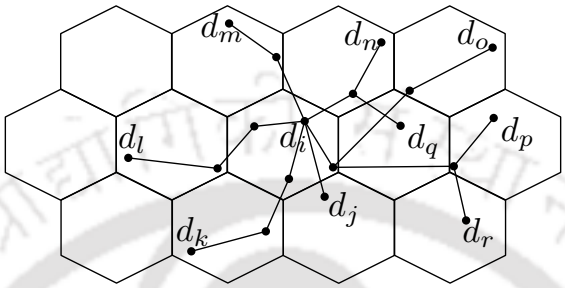


Figure 4.3: Special paths between d_i and other dominators (except d_r) using connectors.

Definition 4.2.6. A triplet of hexagons H_i, H_j, and H_k in the partition are said to be a *linear successive cell triplet* if the centers of the hexagons are collinear and H_j is adjacent to H_i and H_k (see Figure 4.2 (c)). The cell H_j is referred as the *middle cell* of the linear successive cell triplet. H_i and H_k are referred to as the *end cells* of the linear successive cell triplet.

Definition 4.2.7. The *distance* between sets $D', D'' \subseteq \mathcal{D}$ is the minimum hop distance between nodes d_i and d_j, where $d_i \in D'$ and $d_j \in D''$.

Lemma 4.2.8. *Any pair of non-empty complementary subsets of $\mathcal{D}$ are at most 3-hop away from each other. In other words, the distance between any pair of non-empty complementary subsets of $\mathcal{D}$ is at most 3.*

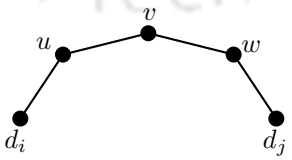


Figure 4.4: Demonstration of Lemma 4.2.8.

Proof. Let D and D'' be two subset of $\mathcal{D}$ such that $D' \cap D'' = \emptyset$ and $D' \cup D'' = \mathcal{D}$. The objective is to show that D' and D'' are at most 3-hop away. On contrary, assume that D' and D'' are at least 4-hop away. Let u, v, and w are the nodes connecting the dominators d_i and d_j, where $d_i \in D'$ and $d_j \in D''$ (see Figure 4.4). Since v is a non-dominator node, it must be dominated by a node in $\mathcal{D}$, say d_k. As $D' \cap D'' = \emptyset$ and $D' \cup D'' = \mathcal{D}$, d_k must be either in D' or D''. If $d_k \in D'$, then d_k and d_j are 3-hop away from each other via v and w. If $d_k \in D''$, then d_i and d_k are 3-hop away from each other via u and v. In either case we arrived at contradiction. Thus the lemma follows. □

Lemma 4.2.9. *For any dominator $d_i \in \mathcal{D}$ ($|\mathcal{D}| > 1$), there exists at least 1 special dominator of d_i in $\mathcal{D}$, i.e., $|SD(d_i)| \geq 1$.*

Proof. Consider the dominator d_i in H_i and an edge $e(u, v) \in G$ such that u is in H_i and v is in H_j, $i \neq j$. Note that such an edge exists in the connected graph G for some j, except for the trivial case where G is entirely within a single cell and $|\mathcal{D}| = 1$. Let d_j be the dominator in H_j. We consider all possible cases as follows: (i) if $u = d_i$ and $v = d_j$, then d_i, d_j is a pair of 1-hop dominators, (ii) if $u = d_i, v \neq d_j$ or $u \neq d_i, v = d_j$, then d_i, d_j is a pair of 2-hop special dominators, and (iii) if $u \neq d_i, v \neq d_j$, then d_i, d_j is a pair of 3-hop special dominators. In all the cases there is always a special dominator of d_i. Thus the lemma follows. □

Corollary 4.2.10. *If (D', D'') is a pair of non-empty complementary subsets of $\mathcal{D}$, then there exists at least one pair of dominators $(d_i, d_j) \in (D', D'')$ such that (d_i, d_j) form a pair of (1, 2, 3) - special dominators, i.e., $d_j \in SD(d_i)$ and also $d_i \in SD(d_j)$.*

Proof. Follows from Lemma 4.2.8 and Lemma 4.2.9. □

Alzoubi et al. [4] first obtained a maximal independent set and considered pairs of nodes that are at most 3-hop away in order to obtain the connector set $\mathcal{C}$. The authors showed that the number of nodes in the maximal independent set that are at most three hops away from an arbitrary node in the maximal independent set is at most 47. In our

algorithm, we consider only pairs of 2-hop special and 3-hop special dominators in $\mathcal{D}$ to obtain $\mathcal{C}$ and get a bound of 18 on the total number of (1, 2, 3) - special dominators from any dominator in $\mathcal{D}$.

Lemma 4.2.11. $|SD(d_i)| \leq 18$ *for any* $d_i \in \mathcal{D}$, *if* $|\mathcal{D}| > 1$.

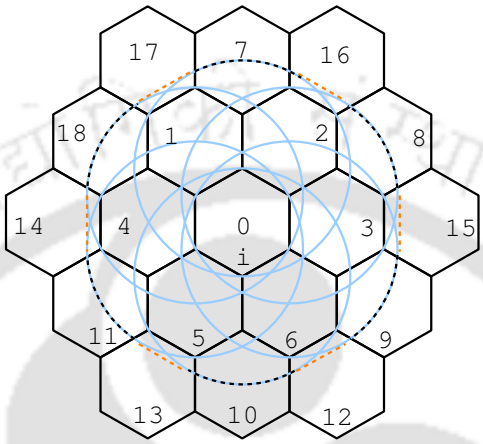


Figure 4.5: The cells within unit distance from H_i.

Proof. Consider a non-empty cell H_i. Let d_i be its dominator. The objective is to show that the cardinality of the set of (1, 2, 3) - special dominators of d_i is at most 18. Recall that (1, 2, 3) - special dominators of d_i are 1-hop, 2-hop special, and 3-hop special dominators. Therefore, the (1, 2, 3) - special dominators of d_i are the dominators lying in the cells that are within a unit distance apart from H_i and there are 18 such cells (see Figure 4.5). Thus the lemma follows. □

To connect d_i to all dominators in $SD(d_i)$, we might have to chose a path between at most 18 dominator pairs (d_i, d_j), where $d_j \in SD(d_i)$. However, using the Lemma 4.2.12, we improve the bound on the number of such dominator pairs that have to be considered to establish connection between d_i to all nodes in $SD(d_i)$.

Lemma 4.2.12. *Let* $d_i \in \mathcal{D}$. *If there is a special path between a pair of nodes in* $SD(d_i)$ *that are in the end cells of a linear successive cell triplet, then the connector node(s) on*

the special path also connect(s) the dominator (if any) in the middle cell of the linear successive cell triplet.

Proof. Consider a linear successive cell triplet consisting of cells H_a, H_b, and H_c, where H_b is the middle cell and H_a and H_c are the end cells. Assuming H_a, H_b, and H_c are to be non-empty, let d_a, d_b, and d_c be their respective dominators. If there is a special path between d_a and d_c (i.e., $d_c \in SD(d_a)$), then there can be at most two connector nodes on the special path. Let u and v be the connector nodes on the special path, where $u \in H_a$, $v \in H_c$, and $e(u, v) \in E$. If there is only one connector node on the special path, then either $u = d_a$ or $v = d_c$. We first prove that any node in H_b is adjacent to at least one of the nodes u or v.

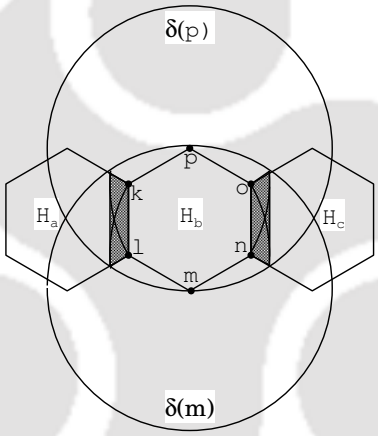


Figure 4.6: Demonstration of Lemma 4.2.12.

The shaded regions shown in Figure 4.6 in H_a (resp. H_c) represents the region that is within unit distance from H_c (resp. H_a) i.e., the region where u (resp. v) lies. Let $k, l, m, n, o,$ and, p be the set of corners of the middle cell in the triplet. Also, let $\delta(u)$, $\delta(v)$, $\delta(p)$, and $\delta(m)$ be the unit disks centered at u, v, p, and m, respectively. Observe that u and v lie in the intersection region of the disks $\delta(p)$ and $\delta(m)$. This implies that, p and m lie inside the disks $\delta(u)$ and $\delta(v)$, respectively. Hence, k and l lie in $\delta(u)$ and o, n lie in $\delta(v)$ and the regular hexagon H_b entirely lies inside $\delta(u) \cup \delta(v)$. Hence, any

node in H_b is adjacent to either u or v. From the above argument, d_b is adjacent to at least one of u or v. Therefore, the connector nodes on the special path between d_a and d_c also connect d_b. □

Corollary 4.2.13. *Let $d_i \in \mathcal{D}$. Special paths between at most 12 dominator pairs (d_i, d_j) such that $d_j \in SD(d_i)$ is sufficient to ensure paths between d_i and each dominator in $SD(d_i)$.*

Proof. Let us consider a cell H_i and its surrounding 18 cells (see Figure 4.5). Let d_i be the dominator in H_i. By Lemma 4.2.11, $|SD(d_i)| \leq 18$. That is, there can be at most 18 (d_i, d_j) (1, 2, 3) - special dominator pairs. In Lemma 4.2.12, we proved that if we can connect d_i with the dominator in the farthest hexagon to H_i in the triplet (where H_i is an end cell), then the connector node(s) also connect(s) the dominator in the adjacent cell of H_i in the triplet. There are six triplets correspond to six sides of H_i. Thus, the connectors on the special paths between d_i and the dominators in the farthest hexagons to H_i in the triplets also connect the dominators in the six adjacent cells of H_i. Therefore, the maximum number of dominator pairs (d_i, d_j) needed to connect all 18 dominators in $SD(d_i)$ is $6 + (18 - 6 \times 2) = 12$. □

4.3 Local Variables

Each node $v \in V$ has a unique ID, denoted by ID(v), and maintains several local variables and arrays. Every node v maintains a local variable My_Dominator which stores the ID and geographic location of its dominator and is denoted by My_Dominator(v). For example, if v is dominator, then My_Dominator(v) stores ID(v) and $(x(v), y(v))$. If v is a non-dominator, then it stores ID(u) and $(x(u), y(u))$, where u is dominator lying in the same cell as that of v. Every node numbers its surrounding 18 cells as shown in Figure 4.5. Every non-dominator v maintains an array DC of size 18. Each entry in DC is an ordered pair, say (d, c), where d is a dominator different from My_Dominator(v)

and c is a node which lies in the same cell as that of d and is adjacent to v. Also, the index of (d, c) corresponds to the cell number in which d lies.

A dominator node v maintains two arrays DCN and DP. The array DCN stores the information in My-Dominator and DC array corresponds to each of its neighbors. Each entry in DCN consists of 3 parts: a neighbor, its dominator, and a pointer pointing to the DC array of the node. Hence, $|DCN| = O(deg(v))$, where $deg(v)$ represents the number of nodes adjacent of v. Each entry in the array DP is an ordered triplet (d_i, c_1, c_2), where d_i is a dominator (different from v), c_1 and c_2 are the connector nodes to d_i such that c_1 lies in the same cell as that of v and c_2 lies in the same cell as that of d_i and is adjacent to c_1. In case v and d_i are 1-hop dominators, then $c_1 = d_i$ and c_2 is null. In case v and d_i are a pair of 2-hop special dominators, then c_2 is null. Also, the index of (d_i, c_1, c_2) corresponds to the numbering of the cell in which d_i lies. Hence, |DP| = 18 (by Lemma 4.2.11).

4.4 Scheduling Scheme

In this section, a scheduling scheme to address the interference problem in WANETs is discussed in detail. The proposed scheduling scheme ensures assignment of time slots to the nodes in order to minimize the interference. As the transmission range of all nodes is identical and equal to 1 unit, the unit disks centered at two nodes whose distance is greater than 2 do not intersect. Therefore, the nodes at a distance greater than 2 units can transmit in the same time slot without causing any interference.

The proposed scheduling scheme is based on the septa-hexagonal partition and its 4-coloring scheme. Consider a septa-hexagon which is assigned color A and its adjacent septa-hexagons are assigned colors B, C and D such that opposite septa-hexagons are assigned the same color. If a point lies on the common boundary of 2 or 3 septa-hexagons, then it belongs to the septa-hexagon whose center has the lowest x coordinate (e.g., in Figure 4.7, points p belongs to the septa-hexagon colored A).

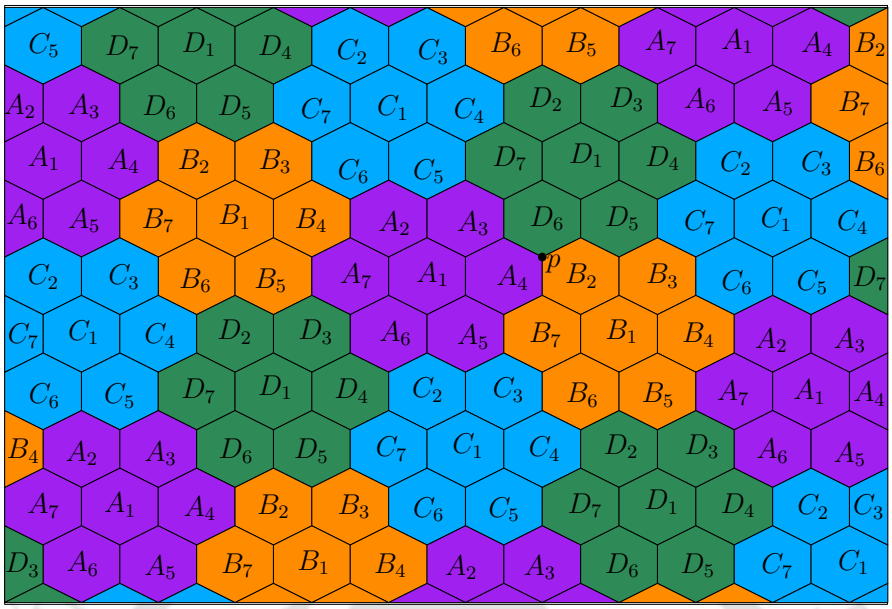


Figure 4.7: A 4-coloring scheme of the hexagonal grid.

Any two points belong to two different septa-hexagons of same color, respectively, are at a distance greater than 2 units. Thus, they can transmit in the same time slot without causing interference. Therefore, we consider division of a time slot into 4 major sub-time slot divisions, assigned one for each color. Each septa-hexagon has 7 cells in it and labeled as shown in Figure 4.7. The sub-time slot assigned to a septa-hexagon is further divided into 7 chunks, one for each cell. The maximum number of nodes in a single cell is Δ (except the trivial case where G entirely lies within a single cell, causing the maximum number of nodes in the cell $\Delta + 1$). We further divide the chunk assigned to a cell into Δ time slots. Thus, we obtain $4 \times 7 \times \Delta = O(\Delta)$ time slots. The assignment of Δ time slots in a cell is based on the increasing order of the node IDs, i.e., a node with smallest ID is assigned an earlier time slot.

Note that, a node can determine the color and labeling of the cell based on its location if the reference to the coloring scheme is known. Since a node has information of its neighbors in its cell, it can determine its time slot to transmit. For example, a node in a cell labeled C_3 whose ID is the 4^{th} lowest in that cell would be assigned the

$(7\Delta + 7\Delta + 2\Delta + 4)^{th}$ time slot of the 28Δ time slots. In the sum, the first time slot 7Δ is for the nodes in the septa-hexagon colored A, the second time slot 7Δ is for the nodes in the septa-hexagon colored B, and the third time slot 2Δ corresponds to the nodes lying in the cells labeled $C1$ and $C2$ of the septa-hexagon colored C.

4.5 Distributed Algorithm

The distributed construction of a CDS can be described in two phases. In the first phase, a dominating set $\mathcal{D}$ is constructed by choosing the lowest ID node, as the dominator, from each non-empty cell. Since each node has a unique ID, it can find the cell in which it lies and its dominator based on geographic location. Each node then broadcasts a message stating its dominator. The distributed algorithm for the first phase is given in Phase 1 in Subsection 4.5.1. In the second phase, special paths are considered between pairs of 2-hop special and 3-hop special dominators. The intermediate nodes (i.e., connector nodes) on these paths form the connector set $\mathcal{C}$. The distributed algorithm for the second phase is given in Phase 2 in Subsection 4.5.1. Since $|SD(d_i)| \leq 18$ for any dominator $d_i \in \mathcal{D}$, the maximum number of distinct 2-hop special and 3-hop special dominator pairs is at most $9|\mathcal{D}|$ (as each dominator can have at most 18 special dominators, and hence there can be up to 18 pairs correspond to a dominator d_i. However, if we consider all the pairs, each pair is being repeated, i.e., (d_i, d_j) comes from $SD(d_i)$ and (d_j, d_i) comes from $SD(d_j)$. So, totally there will be at most $9|\mathcal{D}|$ distinct pairs. In our case the pairs (d_i, d_j) and (d_j, d_i) are the same). Lemma 4.2.12 encourages us to find a subset of these pairs of dominators whose connection would ensure connection of all pairs of dominators (see Lemma 4.5.1). Let us call this subset as SDP and is a subset of the set of $9|\mathcal{D}|$ distinct dominator pairs. We also use $DP(d_i)$ to denote the set of pairs in SDP such that the pair contains d_i as one of its nodes. By Lemma 4.5.2, the cardinality of the set SDP is $6|\mathcal{D}|$. Every dominator node considers itself as d_0 in H_0 and numbers the surrounding 18 cells as shown in Figure 4.5. Note that this numbering is done locally

at each node and is such that (i) H_1 to H_6 are adjacent to H_0, (ii) for $1 \leq i \leq 6$, H_0, H_i and H_{18-i} form a linear successive cell triplet, and (iii) for $1 \leq i \leq 6$, H_i, H_0 and H_{7-i} form a linear successive cell triplet. Every dominator node d_i obtains the set $DP(d_0)$ by considering itself as d_0. Then, we obtain $SDP = \bigcup_{i=1}^{|\mathcal{D}|} DP(d_k)$. We specify two RULES in order to obtain the set $DP(d_0)$ and they are crucial in design of our distributed algorithm.

RULE 1: for $7 \leq i \leq 18$ (cells that are not adjacent to H_0) if $d_i \in SD(d_0)$, then $(d_0, d_i) \in DP(d_0)$.

RULE 2: for $1 \leq i \leq 6$ (cells that are adjacent to H_0) if $d_i \in SD(d_0)$ and $d_{18-i} \notin SD(d_0)$, and (d_i, d_{7-i}) are not a 2-hop special or 3-hop special dominator pair, then $(d_0, d_i) \in DP(d_0)$.

The following lemma says, by obtaining paths between every pair of dominators in SDP, a path between every $9|\mathcal{D}|$ dominator pairs is ensured, where SDP is a subset of the set of $9|\mathcal{D}|$ distinct dominator pairs.

Lemma 4.5.1. *If each pair of dominators in SDP is connected by a path, then a path between every pair of special dominators is ensured.*

Proof. On the contrary assume that there exists a pair d_0, d_i of 2-hop special or 3-hop special dominators which does not have a path connecting each other and $d_i \in SD(d_0)$. Now we consider the following possible two cases.

Case 1 : $7 \leq i \leq 18$

In this case $(d_0, d_i) \in DP(d_0) \subseteq SDP$ (see RULE 1), but there is a path between all dominator pairs in SDP, which leads to a contradiction.

Case 2 : $1 \leq i \leq 6$

In this case either (a) $(d_0, d_i) \notin SDP$ or (b) $d_{18-i} \in SD(d_0)$ or (c) (d_i, d_{7-i}) is a pair of 2-hop special or 3-hop special dominators. For the sub-case (a) the cells H_0, H_i, and H_{18-i} form a linear successive cell triplet and from Lemma 4.2.12, a path between d_0 and d_{18-i} connects d_i to both d_0 and d_{18-i}. For the sub-case (b) $(d_0, d_{18-i}) \in DP(d_0)$ (From RULE 1) and hence there is a path between d_0 and d_{18-i} which connects d_0 and d_i. In the

sub-case (c), since (d_i, d_{7-i}) is a pair of 2-hop special or 3-hop special dominators, H_i, H_0, and H_{7-i} form a linear successive cell triplet and from Lemma 4.2.12, a path between d_i and d_{7-i} connects d_0 to both d_i and d_{7-i}. From RULE 1, $(d_i, d_{7-i}) \in DP(d_i) \subseteq SDP$ and hence there is a path between d_i and d_{7-i} which connects d_0 and d_i. Therefore, in all the sub-cases we arrived at contradiction. Thus the lemma follows. □

Lemma 4.5.2. *Connection between at most* $6|\mathcal{D}|$ *pairs of 2-hop special and 3-hop dominators ensures connection between every pair of 2-hop special and 3-hop special dominators.*

Proof. From Corollary 4.2.13, $|DP(d_i)| \leq 12$ for $1 \leq i \leq |\mathcal{D}|$. Therefore there are at most $12|\mathcal{D}|$ possible pairs of dominators. However, each pair is counted twice. Therefore by Lemma 4.5.1, connection of at most $\frac{12|\mathcal{D}|}{2} = 6|\mathcal{D}|$ pairs of 2-hop special and 3-hop dominators ensures connection between every pair of 2-hop special and 3-hop special dominators. Hence, $|SDP| = 6|\mathcal{D}|$. □

By Lemma 4.5.2, the dominating set $\mathcal{D}$ together with the connector nodes (intermediate nodes) on every path between the dominator pairs in SDP form a CDS.

4.5.1 Distributed implementation

Every node v first executes the algorithm given in Phase 1 to obtain a DS and then Phase 2 to introduce some connector nodes. In Phase 2, we give algorithms for a dominator and non-dominator separately.

Phase 1:

Algorithm for finding a DS:

A node having the lowest ID from each non-empty hexagonal cell broadcasts a message to its neighbors stating that its the dominator of the cell. Let $\mathcal{D}$ be the set of nodes from each non-empty cell having the lowest ID. Therefore, $\mathcal{D}$ is a dominating set by Observation 4.2.1.

After a dominating set has been obtained, each node is either a dominator node or a non-dominator node. Every node broadcasts its dominator information to all of its neighbors via My-Dominator message. By the end of this phase, each node $v \in V$ knows whether it is a dominator node or a non-dominator node, and it also knows dominators of its neighbors.

Phase 2:

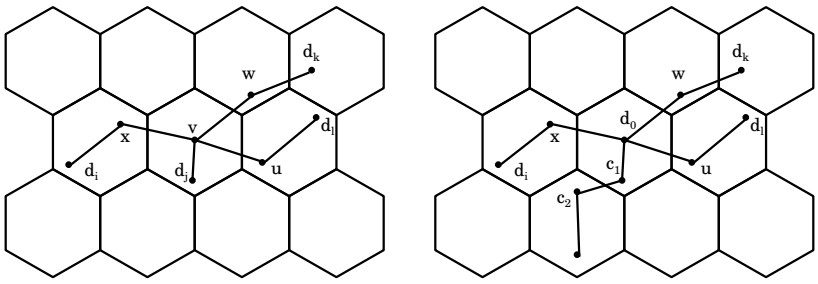
Algorithm for a non-dominator:

Let v be a non-dominator node (i.e., $v \notin \mathcal{D}$) and its dominator be d_0. The dominator d_0 needs the information of up to 2-hop neighbors of v that are dominators (no other information is required). Since the node v can be a connector node in the future and it is at most 2-hop away from a dominator that belongs to a special dominator pair in which the node v plays the role of connector (see Figure 4.8(a)). Therefore, the node v requires information about its neighbor's dominators. This information is got by the My-Dominator messages sent by each neighbor of v. Every non-dominator node obtains its DC array and broadcasts to its neighbors.

Algorithm for a dominator:

Let v be a dominator (i.e., $v \in \mathcal{D}$). It needs the information of up to 3-hop neighbors of v that are dominators. This information is got from its neighbors when they broadcast their DC array. The node v (considered as d_0 hereafter) needs to be connected to all of its special dominators. Potential special dominators to v can be found in the surrounding 18 cells (see Figure 4.5). After receiving all My-Dominator messages from its neighbors, the array DP is obtained. Hence a path is found for all special dominators to d_0 that are 1-hop or 2-hop away.

Now, the dominator d_0 needs to establish paths from d_0 to all special dominators that are at most 3-hop away from d_0. For this, the dominator d_0 requires information about dominators of its 2-hop neighbors (see Figure 4.8(b)). This information is available in the DC array of the non-dominators that are adjacent to d_0 and in the DP array of the dominators that are adjacent to d_0. In this phase each neighbor of d_0 broadcasts a



(a) (b)

Figure 4.8: (a) The dominators (end nodes) which are at most 2-hop away from v, and (b) the dominators (end nodes) which are at most 3-hop away from d_0.

message (DC array in case of non-dominator, the current DP array in case of dominator). It stores the DC message information for each of the neighbor of d_0 in the array DCN. Upon receiving the DC messages from all of its neighbors, the required information is got to establish paths between d_0 and all the special dominators that are at most 3-hop away from d_0. On receiving DC messages from its neighbors, d_0 updates its DCN, DP arrays and applies RULE 1 and RULE 2 to delete some entries in DP.

Note: During the application of RULE 2, for $1 \leq i \leq 7$, the information such as whether d_i and d_{7-i} are a pair of 2-hop special or 3-hop special dominators or not should be known to d_0. If d_i and d_{7-i} are a pair of 2-hop special or 3-hop special dominators, then there exists an edge say $e(a, b)$ such that $a \in H_i$ and $b \in H_{7-i}$. By Lemma 4.2.12, d_0 is adjacent to either a or b. Without loss of generality, assume d_0 is adjacent to a. Now, $(d_{7-i}, b) \in DC(a)$. As d_0 knows DC(a) and My-Dominator(a) from DC and My-Dominator messages from a, it can find out whether d_i and d_{7-i} are a pair of 2-hop special or 3-hop special dominators or not. After receiving DC messages from all of its neighbors, a dominator broadcasts its DP. A non-dominator is a connector node if it is in the DP of any DP message it receives.

4.5.2 Proof of correctness

Theorem 4.5.3. *The set $\mathcal{M}$ obtained in Subsection 4.5.1 is a CDS of G.*

Proof. On contrary, assume that $\mathcal{M}$ obtained in Subsection 4.5.1 is not a CDS of G. Let

$\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k$ be the disjoint components of $\mathcal{M}$, where $k > 1$. Now, for $1 \leq i \leq k$, consider the set of dominator nodes in $\mathcal{M}_i$ and call this set $\mathcal{D}_i$. A component cannot have only connector nodes as they are connected to at least one dominator node. Therefore $\mathcal{D}_i \neq \emptyset$ for all i. Sets $\mathcal{D}_1$ and $\bigcup_{i=2}^{k} \mathcal{D}_i$ are a pair of non-empty complementary subsets of $\mathcal{D}$ and from Lemma 4.2.10, there exists a pair of dominator nodes (d_i, d_j) such that (d_i, d_j) form a pair of (1, 2, 3) - special dominators. If they are 1-hop neighbors, then the number of components in $\mathcal{M}$ is less than k, which leads to a contradiction. In the construction of CDS every pair of 2-hop special and 3-hop special dominators are connected. Hence, the number of components in $\mathcal{M}$ is less than k, arrived at a contradiction. Also, every dominator is in (1, 2, 3) - special dominator of some other dominator (see Lemma 4.2.9). Therefore, all dominators are connected in $\mathcal{M}$. So $\mathcal{M}$ is a CDS of G. □

4.5.3 Approximation factor

We obtain a bound on the size of the CDS $\mathcal{M}$ by observing some geometric properties of unit disk graphs.

Lemma 4.5.4. *A single unit disk centered at any point inside a cell cannot cover points in more than 12 cells simultaneously [34].*

Lemma 4.5.5. *The intersection region of two unit radius disks centered at two adjacent points covers some points in at least 4 cells.*

Proof. Consider two adjacent points p and q and the unit radius disks $\delta(p)$ and $\delta(q)$ centered at p and q, respectively. Let c and d be the intersection points of the disks $\delta(p)$ and $\delta(q)$. Let a (resp. b) be the point of intersection of the boundary of $\delta(q)$ (resp. $\delta(p)$) and the line segment joining p and q. Let the intersection point of the line segments $\overline{cd}$ and $\overline{ab}$ be o. Consider a disk $\delta(o)$ centered at o with radius $d(o, a)$ (see Figure 4.9). Observe that $d(o, a) \geq \frac{1}{2}$ and $\delta(o)$ lies inside the region $\delta(p) \cap \delta(q)$. Let S be the set of corners of the regular hexagon(s) in the intersection region of $\delta(p)$ and $\delta(q)$.

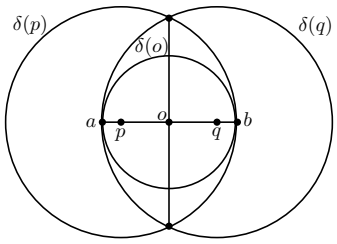


Figure 4.9: Demonstration of Lemma 4.5.5.

The following two cases may arise:

Case 1 ($o \notin S$): Consider the cell o lies in and let us number it as 1 (the adjacent cells numbered 2 to 7) and its corners as g, h, i, j, k, and l (see Figure 4.10). We divide the cell numbered 1 into 6 equilateral triangles of side length $\frac{1}{2}$ namely A, B, C, D, E, and F (see Figure 4.10). The center o can lie in any one of the 6 sub-cells. Without loss of generality assume that o lies in A and $o \neq g, h, m$. Note that, $d(o, g) < \frac{1}{2}$ and $d(o, h) < \frac{1}{2}$, and the side $\overline{gh}$ completely lies in $\delta(o)$. That is, the region A completely lies in $\delta(o)$. Thus, $\delta(o)$ covers some points in the cells 1, 2, 3, and 4.

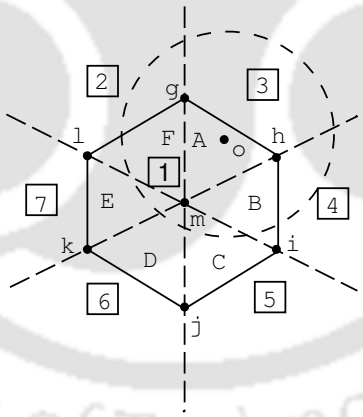


Figure 4.10: Demonstration of Case 1.

If o lies at the center of the cell (call it m), then it covers some points in the cells numbered 1 to 7. Thus, a disk of radius $\geq \frac{1}{2}$ centered at o covers some points in at least 4 cells.

Case 2 ($o \in S$): Suppose o lies at a corner of a hexagon inside the intersection of the disks $\delta(p)$ and $\delta(q)$. Let r, s, t be the corners adjacent to that corner. As the side length of a hexagon is $1/2$, the corners r, s, t lie on or within the boundary of $\delta(o)$ (see Figure 4.11).

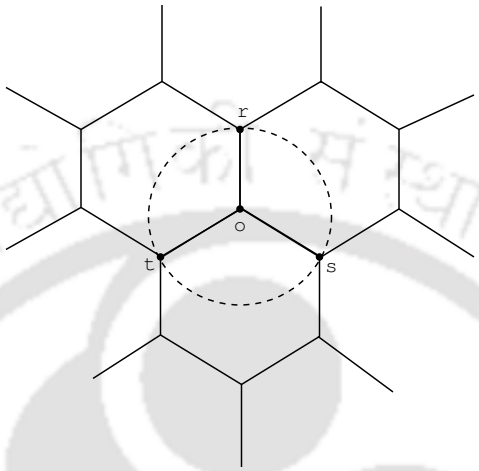


Figure 4.11: Demonstration of Case 2.

Note that at least one of r, s, t is completely inside $\delta(p) \cap \delta(q)$ as the radius of the disk $d(o)$ is greater than or equal to $\frac{1}{2}$. Without loss of generality, say r is completely inside $\delta(p) \cap \delta(q)$, which implies the side $\overline{or}$ is completely inside $\delta(p) \cap \delta(q)$. Thus $\delta(p) \cap \delta(q)$ is partitioned into at least 4 parts. Hence, $\delta(p) \cap \delta(q)$ covers some points in at least 4 cells. Thus the lemma follows. □

Lemma 4.5.6. *The size of the set $\mathcal{D}$ is at most $8opt + 4$, where opt represents the size of an MCDS of G.*

Proof. Consider an MCDS of G of size *opt* and a depth-first-search traversal $v_1, v_2, \ldots, v_{opt}$ of any spanning tree of the MCDS. We partition $\mathcal{D}$ into $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_{opt}$, where $\mathcal{D}_1$ represents the set of dominators adjacent to v_1 and for $2 \leq i \leq opt$, $\mathcal{D}_i$ represents the set of remaining dominators that are adjacent to v_i but none of $v_1, v_2, \ldots, v_{i-1}$. Let $\delta(v_i)$ represent the unit disk centered at v_i, $1 \leq i \leq opt$. Observe that the dominators in $\mathcal{D}_i$ lie in

the region $Z_i = \delta(v_i) \setminus \bigcup_{j=1}^{i-1} \delta(v_j)$. More than one Z_k can cover some points in a single cell. However the dominator of that cell belongs to only one of the regions. We can safely assume that the dominator is in Z_ℓ, where $\ell = min\{k : Z_k \text{ covers some points in the cell}\}$, as such assumption doesn't change the total sum of the number of dominators. By Lemma 4.5.4, $|\mathcal{D}_1| \leq 12$. Consider a node v_i, where $2 \leq i \leq opt$, $\delta(v_i)$ can cover some points from at most 12 cells as so is Z_i. However, by Lemma 4.5.5, dominators of at least 4 of those 12 cells have already been considered by $\bigcup_{j=1}^{i-1} \delta(v_j)$ as v_i is adjacent to at least one of $v_1, v_2, \ldots, v_{i-1}$. Thus Z_i has at most 8 dominators in its region, i.e. $|\mathcal{D}_i| \leq 8$ for $2 \leq i \leq opt$. Therefore, $|\mathcal{D}| = \sum_{i=1}^{opt} |\mathcal{D}_i| = |\mathcal{D}_1| + \sum_{i=2}^{opt} |\mathcal{D}_i| \leq 12 + 8 \times (opt - 1) \leq 8opt + 4$. □

Theorem 4.5.7. *The size of the CDS $\mathcal{M}$ generated by the algorithm in Subsection 4.5.1 is at most $104opt + 52$, where opt is the size of an MCDS of G.*

Proof. At most $2 \times 6|\mathcal{D}| = 12|\mathcal{D}|$ connectors are needed to connect the dominator nodes in $\mathcal{D}$ (by Lemma 4.2.8 and Lemma 4.5.2), i.e., $|\mathcal{C}| \leq 2 \times 6|\mathcal{D}|$. Thus, the total number of nodes in the CDS $\mathcal{M}$ generated by the algorithm is at most $|\mathcal{D}| + 12|\mathcal{D}|$. Therefore, $|\mathcal{M}| \leq |\mathcal{D}| + 12|\mathcal{D}| = 13|\mathcal{D}| \leq 13 \times (8opt + 4) \leq 104opt + 52$ □

4.5.4 Time and message complexities

Theorem 4.5.8. *The time and message complexities of our algorithm is $O(\Delta)$ and $O(n)$, respectively, where Δ is the maximum node degree in G and n is the number of nodes in G.*

Proof. For every node executing the distributed algorithm, the first phase of the algorithm takes $O(\Delta)$ as the number of nodes in a hexagonal cell is at most $O(\Delta)$. Each non-dominator takes $O(\Delta)$ time to build its DC array after receiving My-Dominator message from all of its neighbors. The dominator takes $O(\Delta)$ time to build its DCN and DP arrays upon receiving DC messages from all of its neighbors. Each node executes the distributed algorithm in parallel. Thus, the time complexity of our algorithm is O(Δ). By explicitly dealing with interference in the proposed scheduling scheme, each node

has to wait for at most $O(\Delta)$ time slots to receive messages from all of its neighbors. Therefore the time complexity of our distributed algorithm is $O(\Delta)$.

Since each node sends a constant number of messages, the total number of messages is $O(n)$ and thus the message complexity of our algorithm is $O(n)$. □

4.6 Conclusion

We proposed a constant factor distributed approximation algorithm for the MCDS problem on a UDG corresponds to a given set $\mathcal{P}$ of n points in $\mathbb{R}^2$. The running time of the proposed algorithm is $O(\Delta)$, where Δ is the degree of the UDG. The proposed algorithm outperforms the existing algorithms in the literature in terms of running time. Another advantage of our algorithm is low message complexity (linear in n). We also proposed a scheduling scheme that obtains $O(\Delta)$ conflict-free time slots to deal with interference.

Chapter 5

Geometric Minimum Liar's Dominating Set Problem

In this chapter, we study the geometric minimum liar's dominating set problem, defined as follows:

> Geometric minimum liar's dominating set (GMLDS) problem : *Given a set $\mathcal{P}$ of n points in $\mathbb{R}^2$, find a subset D of $\mathcal{P}$ of minimum cardinality satisfying the following two conditions: (i) for each point $p_i \in \mathcal{P}$ there exist at least two points in D which are at distance at most one from p_i, and (ii) for every distinct pair of points p_i and p_j in $\mathcal{P}$, D contains at least three points from the closed neighborhood union of p_i and p_j.* We call a subset satisfying the above two conditions as geometric liar's dominating set (or simply a liar's dominating set) of $\mathcal{P}$.

5.1 Preliminaries

Let $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ be a set of n points in the plane. Let $d(p_i, p_j)$ denotes the Euclidean distance between p_i and p_j. For $p_i \in \mathcal{P}$, the closed neighborhood of p_i is denoted by $N[p_i]$ and is defined as $N[p_i] = \{p_j \in \mathcal{P} \mid d(p_i, p_j) \leq 1\}$. We define $\Delta = \max\{|N[p_i]| : p_i \in \mathcal{P}\}$. For any two points p_i, p_j in $\mathcal{P}$, if $p_j \in N[p_i]$, then we say

that p_j is a neighbor of p_i (some times we say that p_j is covered by p_i) and vice versa. Since for any liar's dominating set $\mathcal{P}'$, $|(N[p_i] \cup N[p_j]) \cap \mathcal{P}'| \geq 3$ for all $p_i, p_j \in \mathcal{P}$, we assume that $|\mathcal{P}| \geq 3$ and $|N[p_i]| \geq 2$ for each $p_i \in \mathcal{P}$.

The objective of the GMLDS problem is to find a minimum size subset $\mathcal{P}'$ of $\mathcal{P}$ such that: (i) $|N[p_i] \cap \mathcal{P}'| \geq 2$ for each $p_i \in \mathcal{P}$, and (ii) $|(N[p_i] \cup N[p_j]) \cap \mathcal{P}'| \geq 3$ for every distinct p_i and p_j in $\mathcal{P}$. In other words, for a given set of unit disks having centers at the points in $\mathcal{P}$, the goal is to find a subset $\mathcal{P}'$ of these disks such that, (i) the unit disk centered at each point of $\mathcal{P}$ is pierced by at least two disks in $\mathcal{P}'$, and (ii) the unit disks centered at any two points of $\mathcal{P}$ are pierced by at least three disks in $\mathcal{P}'$.

Observe that the underlying graph of the point set $\mathcal{P}$ is a UDG. That is, the point set $\mathcal{P}$ can be treated as a set of n centers of disks of unit radii in the plane and hence a unit disk graph can be constructed. The vertex set in the graph represents $\mathcal{P}$ and an edge $e(p_i, p_j)$ implies that the points p_i and p_j lie in the intersection region of the unit disks centered at p_i and p_j respectively. Let the UDG constructed be $G = (V, E)$, where $V = \mathcal{P}$ and $E = \{e(p_i, p_j) \mid d(p_i, p_j) \leq 1\}$. Now the problem of finding a minimum liar's dominating set of $\mathcal{P}$ is equivalent to finding a minimum liar's dominating set in the UDG $G = (V, E)$.

We use $d_G(u, v)$ to denote the number of edges on a shortest path between u and v in G. The r-th neighborhood of a vertex $v \in V$ is defined as $N^r[v] = \{u \in V \mid d_G(u, v) \leq r\}$. For $A \subseteq V$, $LD(A)$ denotes a liar's dominating set and $LD_{opt}(A)$ denotes an optimal liar's dominating set of A in G. For $A, B \subseteq V$, $d_G(A, B)$ denotes the distance between A and B and is defined as $d_G(A, B) = \min_{a \in A, b \in B} \{d_G(a, b)\}$. For a vertex $v \in V$, we define $N_G[v] = \{u \in V \mid (v, u) \in E\} \cup \{v\}$, the closed neighborhood of v in G. Similarly, we can define the closed neighborhood of a set $A \subseteq V$ i.e., $N_G[A] = \bigcup_{v \in A} N_G[v]$.

We first prove that the GMLDS problem is NP-hard for unit disk graphs. Unlike in general graphs, the GMLDS problem admits constant factor approximation algorithms. We present a simple $\frac{63}{2}$-factor approximation algorithm in $O(n \log n)$ time, followed by a $\frac{732}{k}$-factor approximation algorithm in $O(n^{k+1}\Delta)$ time for $3 \leq k \leq 183$. We then

extend the idea of $\frac{732}{k}$-factor approximation algorithm to get a $\frac{846}{k}$-factor approximation algorithm in $O(n^{k+1}\Delta)$ time for $3 \leq k \leq 282$. Finally, we propose a PTAS for the same problem.

5.2 Hardness of the GMLDS Problem

In this section, we show that the MLDS problem on UDGs is NP-complete by reducing the *vertex cover* problem defined on planar graphs to it, which is known to be NP-complete [48]. The decision versions of both the problems are defined below.

The MLDS problem on UDGs (LDS-UDG)

Instance: A unit disk graph $G = (V, E)$ and a positive integer k.

Question: Does there exist a liar's dominating set L in G such that $|L| \leq k$?.

The vertex cover problem on planar graphs (VC-PLA)

Instance: An undirected planar graph G with maximum degree 3 and an integer $k > 0$.

Question: Does there exist a vertex cover D of G such that $|D| \leq k$?.

To prove the hardness of the problem we use the concept *planar embedding of planar graphs*.

Lemma 5.2.1 ([95]). *A planar graph $G = (V, E)$ with maximum degree 4 can be embedded in the plane using $O(|V|^2)$ area in such a way that its vertices are at integer coordinates and its edges are drawn so that they are made up of line segments of the form $x = i$ or $y = j$, for integers i and j.*

This kind of embedding is known as *orthogonal drawing* of a graph. Biedl and Kant [12] gave a linear time algorithm that produces an orthogonal drawing of a given graph with the property that the number of bends along each edge is at most 2 (see Fig. 5.1).

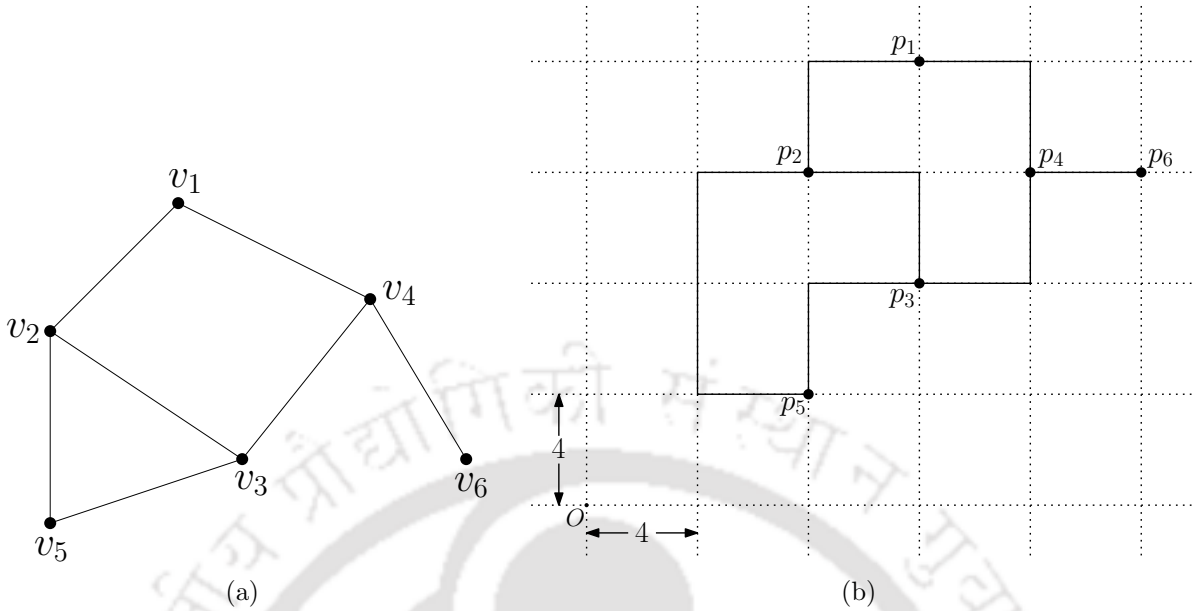


Figure 5.1: (a) A planar graph G with maximum degree 3, and (b) an embedding of G on a grid in the plane.

Corollary 5.2.2. *A planar graph $G = (V, E)$ with maximum degree 3 and $|E| \geq 2$ can be embedded in the plane such that its vertices are at integer coordinates $(4i, 4j)$ and its edges are drawn as a sequence of consecutive line segments on the lines $x = 4i$ or $y = 4j$, for integers i and j.*

In summary, we can embed a planar graph $G = (V, E)$ with maximum degree 3 and $|E| \geq 2$ on a grid in the plane, where each grid cell is of size 4×4, such that

1. Each vertex v_i in V is correspond to a point p_i in the plane.

2. The coordinate of each point p_i (corresponding to a vertex in G) is $(4i, 4j)$ for some integers i and j (see Figure 5.1).

3. Each edge is represented as a sequence of consecutive line segments and is drawn on the lines $x = 4i$ and/or $y = 4j$ for some integers i or j (these consecutive line segments may turn at some positions of the form $(4i', 4j')$). For example, see the edges $e(v_2, v_5)$ and $e(v_1, v_4)$ in Figure 5.1. The former is drawn as a sequence of

four line segments (one horizontal, two vertical, and one horizontal) and the letter is drawn as a sequence of one horizontal and one vertical line segments in the embedding.

4. No two set of consecutive line segments correspond to two distinct edges of G have a common point unless the edges are incident to some vertex of G.

Lemma 5.2.3. *Let $G = (V, E)$ be an instance of* Vc-Pla *with $|E| \geq 2$. An instance $G' = (V', E')$ of* Lds-Udg *can be constructed from G in polynomial-time.*

Proof. We construct G' in four phases.

Phase 1: Embedding of G into a grid of size 4×4

Embed the instance G in the plane as discussed previously using one of the algorithms in [56, 59]. An edge in the embedding is a sequence of connected line segment(s) of length four units each. If the total number of line segments used in the embedding is ℓ, then the sum of the lengths of the line segments is 4ℓ as each line segment has length 4 units. We name the points in the embedding correspond to the vertices of G by *node points*.

Phase 2: Adding extra points to the embedding

Divide the set of line segments in the embedding into two categories, namely, proper and improper. We call a line segment *proper* if none of its end points corresponding to a vertex in G. A line segment is *improper* if it is not a proper segment. For each edge $e(p_i, p_j)$ of length 4 units we add two points at distances 1 and 1.5 units of p_i and p_j, respectively (thus adding four points in total, see the edge $e(p_4, p_6)$ in Figure 5.2(a)). For each edge of length greater than 4 units, we also add points as follows: for each improper line segment we add four points at distances 1, 1.5, 2.5, and 3.5 units from the end point corresponding to a vertex in G, and for each proper line segment we add four points at distances 0.5 and 1.5 units from its end points (see Figure 5.2(a)). We name the points added in this phase *joint points*.

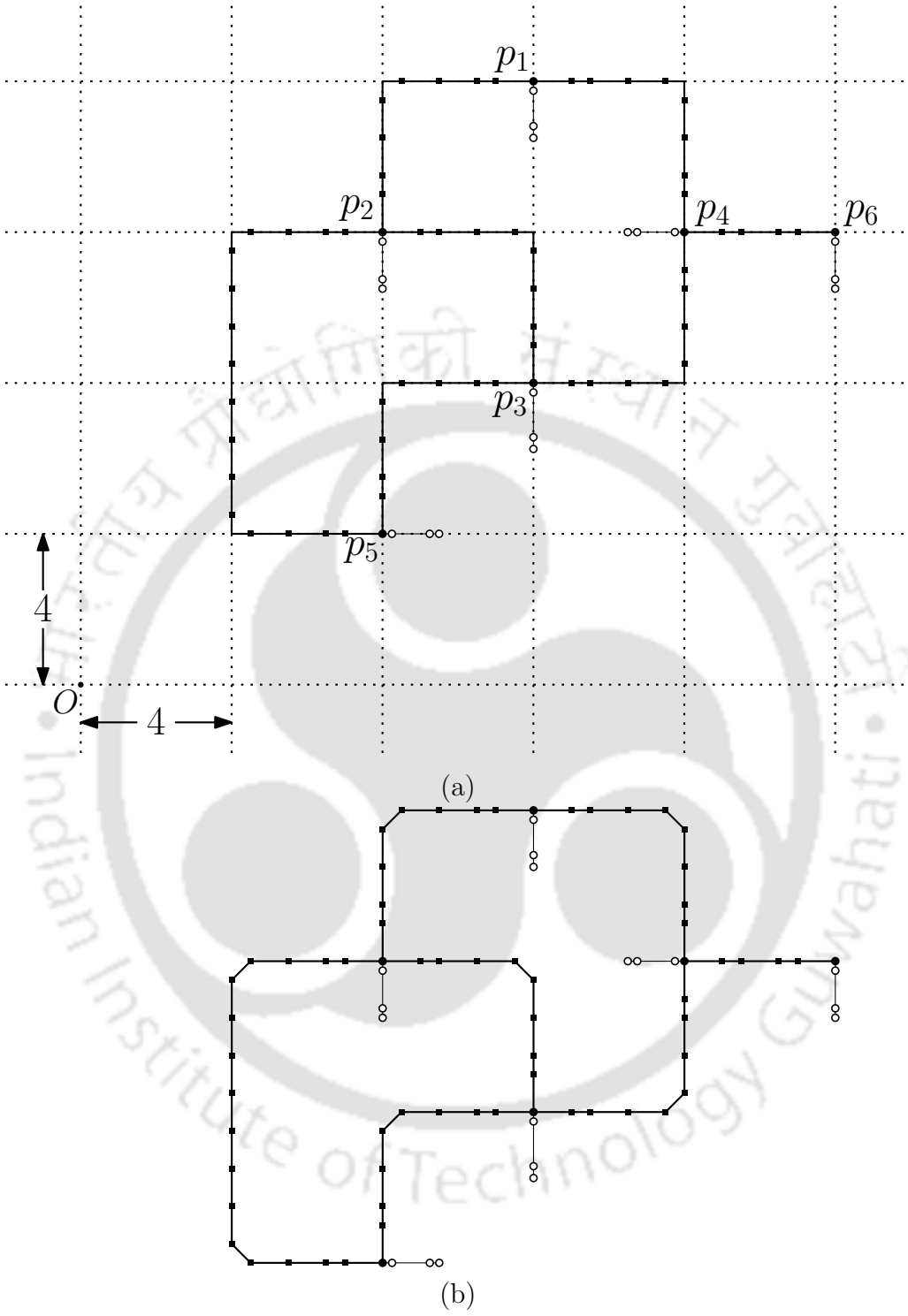


Figure 5.2: (a) Construction of unit disk graph from the embedding, and (b) its corresponding unit disk graph. The node points are represented by •, the joint points are represented by ■, and the support points are represented by ∘.

Phase 3: Adding extra line segments and points

Add a line segment of length 1.4 units (on the lines $x = 4i$ or $y = 4j$ for some integers i or j) for every point p_i (as shown in Figure 5.2(a)) corresponds a vertex v_i in G without coinciding with the line segments that had already been drawn. Observe that adding this line segment on the lines $x = 4i$ or $y = 4j$ is possible without losing the planarity as the maximum degree of G is 3. Now, add three points (say x_i, y_i, and z_i) on these line segments at distances 0.2, 1.2, and 1.4 units, respectively, from the end point corresponds to a vertex in G. We name the points added in this phase *support points*.

Phase 4: Construction of UDG

For convenience, let us denote the set of node points, joint points, and support points by N, J, and S, respectively. In Figure 5.2(a) these sets of points are represented as a set of solid circles (•), solid squares (■), and circles (○), respectively. Let $N = \{p_i \mid v_i \in V\}$, $J = \{q_1, q_2, \ldots, q_{4\ell}\}$, and $S = \{x_i, y_i, z_i \mid v_i \in V\}$. Now we construct a UDG $G' = (V', E')$, where $V' = N \cup J \cup S$ and there is an edge between two points in V' if and only if the Euclidean distance between the points is at most 1 (see Figure 5.2(b)). Observe that, $|N| = |V|(= n)$, $|J| = 4\ell$, where ℓ is the total number of line segments in the embedding, and $|S| = 3|V|(= 3n)$. Hence, $|V'| = 4(n + \ell)$ and ℓ is bounded by a polynomial of n. Therefore G' can be constructed in polynomial-time. □

Theorem 5.2.4. LDS-UDG *is NP-complete.*

Proof. For any given set $L \subseteq V$ and a positive integer k, we can verify whether L is a liar's dominating set of size at most k or not in polynomial-time.

We prove the hardness of LDS-UDG by reducing VC-PLA to it. Let $G = (V, E)$ be an instance of VC-PLA. Construct an instance $G' = (V', E')$ of LDS-UDG as discussed in Lemma 5.2.3. We now prove the following claim: *G has a vertex cover of size at most k if and only if G' has a liar's dominating set of size at most $k + 3\ell + 3n$.*

Necessity: Let $D \subseteq V$ be a vertex cover of G such that $|D| \leq k$. Let $N' = \{p_i \in N \mid v_i \in D\}$, i.e., N' is the set of vertices in G' corresponding to the vertices in D. From each segment in the embedding we chose 3 vertices. The set of chosen vertices, say $J'(\subseteq J)$,

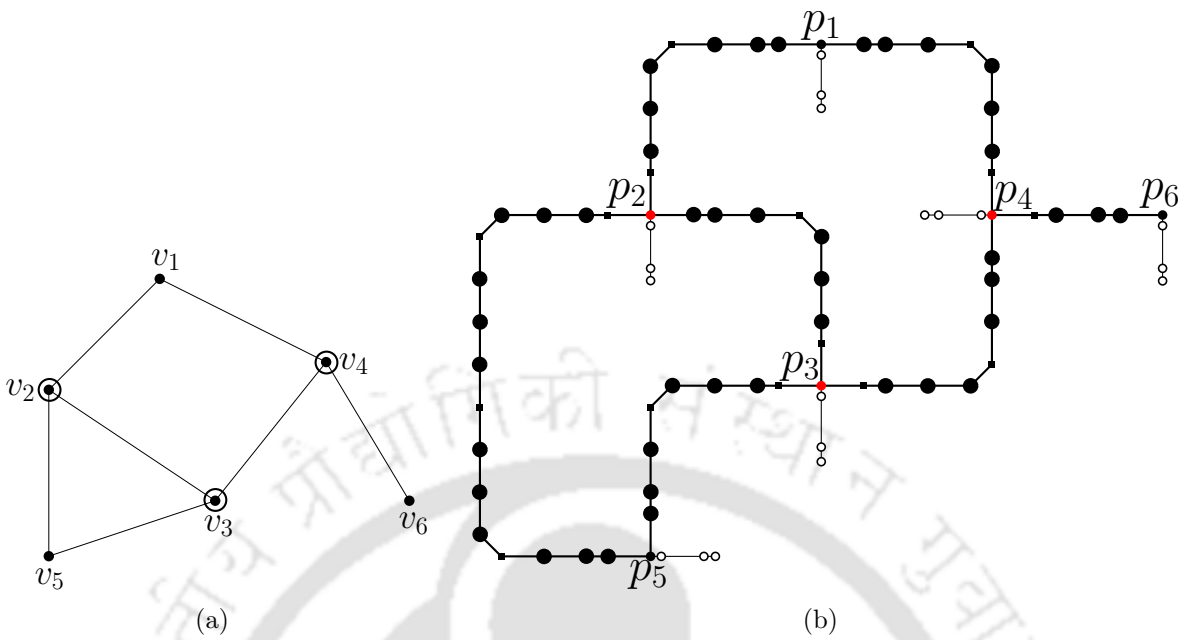


Figure 5.3: (a) A vertex cover $\{v_2, v_3, v_4\}$ in G, and (b) the construction of J' in G' (the tie between v_2 and v_3, and v_3 and v_4 is broken by choosing v_3)

together with N' and S will form an LDS of desired cardinality in G'. We now discuss the process of obtaining the set J'. Initially $J' = \emptyset$. As D is a vertex cover, every edge in G has at least one of its end vertices in D. Let $e(v_i, v_j)$ be an edge in G and $v_i \in D$ (the tie can be broken arbitrarily in case v_i and v_j both are in D). Note that the edge $e(v_i, v_j)$ is represented as a sequence of line segments in the embedding. Start traversing the segments (of $e(v_i, v_j)$) from p_i, where p_i corresponds to v_i, and add all the vertices to J' except the first one from each segment encountered (see (p_2, p_5) in Figure 5.3(b)). Apply the above process to each edge in G. Observe that the cardinality of J' is 3ℓ as we have chosen 3 vertices from each segment in the embedding. Let $L = N' \cup J' \cup S$. Now, we argue that L is a liar's dominating set in G'.

1. Each $p_i \in N$ is dominated by x_i in S. If $p_i \in N'$ (i.e., the corresponding vertex $v_i \in D$ in G), then $|N_{G'}[p_i] \cap L| \geq |\{p_i, x_i\}| = 2$. If $p_i \notin N'$, then there must exists at least one vertex q_j in J' dominating p_i. The existence of q_j is guaranteed by the way we constructed J'. Hence, $|N_{G'}[p_i] \cap L| \geq |\{q_j, x_i\}| = 2$. In either case

every vertex in N is dominated by at least two vertices in L. It is needless to say, every vertex in J is dominated by at least two vertices in $N' \cup J'$. Similarly, every vertex in S is dominated by itself, by its neighbor(s) in S, and, perhaps, by one vertex in N'. Therefore, every vertex in V' is double dominated by vertices in L.

2. Consider every pair of distinct vertices in V'. Of course, every pair of distinct vertices in S satisfy the liar's second condition. We prove that remaining pairs of distinct vertices also satisfy the liar's second condition by considering all possible cases.

   Case a. $p_i, p_j \in N$**:** If at least one of p_i, p_j belongs to N' (without loss of generality say $p_i \in N'$), then $|(N_{G'}[p_i] \cup N_{G'}[p_j]) \cap L| \geq |\{x_i, x_j, p_i\}| = 3$. If none of p_i, p_j belong to N', then there must exists some $q_i, q_j \in J'$ such that q_i, q_j dominate p_i, p_j, respectively. Hence, $|(N_{G'}[p_i] \cup N_{G'}[p_j]) \cap L| \geq |\{x_i, x_j, q_i, q_j\}| = 4$.

   Case b. $q_i, q_j \in J$**:** If both $q_i, q_j \in J'$, then it is trivial that $|(N_{G'}[q_i] \cup N_{G'}[q_j]) \cap L| \geq 3$. Suppose one of q_i, q_j belongs to J' (without loss of generality say $q_i \in J'$). As every vertex in G' is double dominated, q_j must be dominated by two vertices in J' or by either some q_k in J' and some p_l in N'. In either case we get $|(N_{G'}[q_i] \cup N_{G'}[q_j]) \cap L| \geq 3$. The similar argument works even if none of q_i, q_j belong to J'.

   Case c. $p_i \in N$ **and** $q_j \in J$**:** If none of p_i and q_j belong to L, then the argument is trivial as each one is dominated by at least two vertices in L. If both belong to L, then $|(N_{G'}[p_i] \cup N_{G'}[q_j]) \cap L| \geq |\{p_i, x_i, q_j\}| = 3$. If $p_i \in L$ and $q_j \notin L$ (the other case is similar), then $|(N_{G'}[p_i] \cup N_{G'}[q_j]) \cap L| \geq 3$ holds as q_j is double dominated.

   Likewise, we can argue for other pair combinations too. Therefore, every pair of distinct vertices in V' is dominated by at least 3 vertices in L.

Thus, we conclude that L is an LDS in G' and $|L| = |N'| + |J'| + |S| \leq k + 3\ell + 3n$.

Sufficiency: Let $L \subseteq V'$ be an LDS of size at most $k + 3\ell + 3n$. We prove that G has a vertex cover of size at most k with the aid of the following claims.

(i) $S \subset L$.

Proof. The proof directly follows from the definition of liar's dominating set. Observe that we added points x_i, y_i, z_i such that p_i is adjacent to x_i, x_i is adjacent to y_i and y_i is adjacent to z_i in G', i.e., $\{e(p_i, x_i), e(x_i, y_i), e(y_i, z_i)\} \subset E'$, for each i. Hence, z_i and y_i must be in L due to the first condition of liar's domination. Also, every connected component of L in G must contain at least three vertices due to the second condition of liar's domination. Hence, $x_i \in L$. Therefore, any liar's dominating set of G' must contain $\{x_i, y_i, z_i\}, 1 \leq i \leq n$, i.e., $S \subset L$.

(ii) Every segment in the embedding must contribute at least 3 vertices to L and hence $|J \cap L| \geq 3\ell$, where ℓ is the total number of segments in the embedding.

Proof. The proof follows from the fact that only consecutive points are adjacent (in G') on any segment in the embedding. Let η be a segment in the embedding having vertices q_i, q_{i+1}, q_{i+2}, and q_{i+3}. On contrary, assume that η has only two of its vertices in L. Note that both q_{i+1} and q_{i+2} can not be in L simultaneously. If both are present in L, then they do not satisfy the second condition as q_i and q_{i+3} are not in L, i.e., $|(N_{G'}[q_{i+1}] \cup N_{G'}[q_{i+2}]) \cap L| = |\{q_{i+1}, q_{i+2}\}| = 2$; contradiction to L is an LDS. Without loss of generality we assume that $q_{i+2} \notin L$ (the similar argument works when $q_{i+1} \notin L$.). If q_i and q_{i+1} are in L, then $|(N_{G'}[q_{i+1}] \cup N_{G'}[q_{i+2}]) \cap L| = |\{q_i, q_{i+1}\}| = 2$. If q_i and q_{i+3} are in L, then $|(N_{G'}[q_{i+1}] \cup N_{G'}[q_{i+2}]) \cap L| = |\{q_i, q_{i+3}\}| = 2$. If q_{i+1} and q_{i+3} are in L, then $|(N_{G'}[q_{i+1}] \cup N_{G'}[q_{i+2}]) \cap L| = |\{q_{i+1}, q_{i+3}\}| = 2$. In either case we arrived at a contradiction.

(iii) If p_i and p_j correspond to end vertices of an edge $e(v_i, v_j)$ in G, and both p_i, p_j are not in L, then there must be at least $3\ell' + 1$ vertices in L form the segment(s) representing the edge $e(v_i, v_j)$, where ℓ' is the number of segments representing the edge $e(v_i, v_j)$ in the embedding.

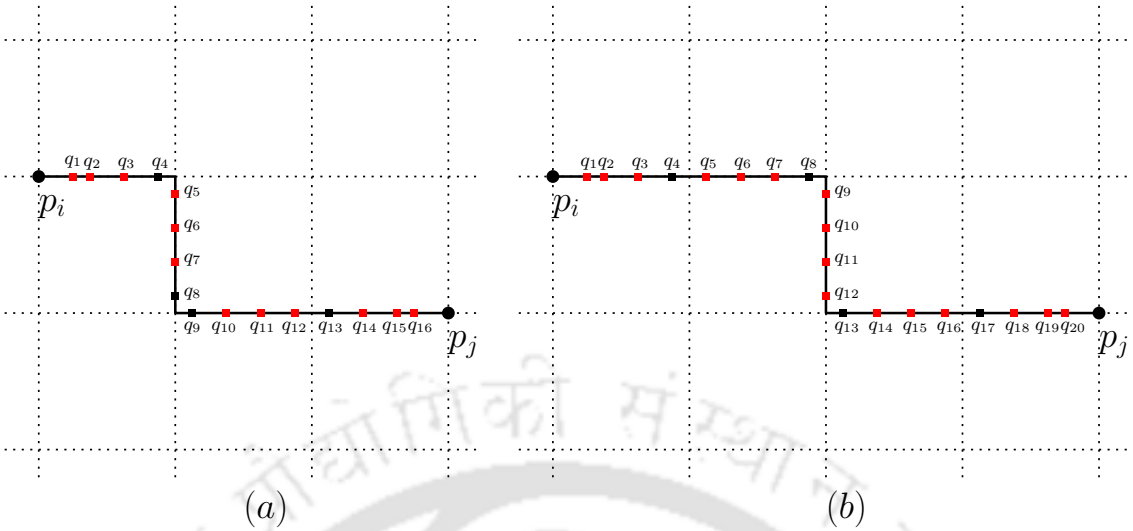


Figure 5.4: Illustration of Claim (iii). The vertices marked red must be in L.

Proof. The proof follows from Claim (ii). Let $e(v_i, v_j)$ be an edge in G such that p_i and p_j are not in L. By Claim (ii) every segment must contribute at least three vertices to L. Hence, the number of vertices in L from the segments representing the edge $e(v_i, v_j)$ is at least $3\ell'$. We argue that if both p_i and p_j are not in L, then the number of vertices in L from the segments representing the edge $e(v_i, v_j)$ is at least $3\ell' + 1$. Suppose that there are exactly $3\ell'$ vertices in L from the segments. That is, no segment representing the edge $e(v_i, v_j)$ contains more than three vertices in L. Let $p_i, q_1, q_2, \ldots, q_{4\ell'}, p_j$ be the vertices encountered while traversing the segments from p_i. If $\ell' = 1$, the argument can be proven as in the proof of Claim (ii). Assume $\ell' > 1$.

Case a. ℓ' **is even:** Since p_i and p_j are not in L and due to the second condition of liar's domination, the vertices q_1, q_2, q_3 from the first segment and $q_{4\ell'-2}, q_{4\ell'-1}, q_{4\ell'}$ from the last segment must be in L. The vertices q_4 and $q_{4\ell'-3}$ can not be in L as we assumed that each segment contains exactly three vertices in L. If we continue in the same manner for the rest of the segments from both sides, we end up in not choosing the vertices $q_{2\ell'}$ and $q_{2\ell'+1}$ from the $\frac{\ell'}{2}$-th and $(\frac{\ell'}{2} + 1)$-th segments, respectively. Note that $q_{2\ell'}$ is

the last vertex on $\frac{\ell'}{2}$-th segment and $q_{2\ell'+1}$ is the first vertex on $(\frac{\ell'}{2}+1)$-th segment and $(q_{2\ell'}, q_{2\ell'+1})$ is an edge in G' (see Fig. 5.4(a)). Also, note that $|(N_{G'}[q_{2\ell'}] \cup N_{G'}[q_{2\ell'+1}]) \cap L| = |\{q_{2\ell'-1}, q_{2\ell'+2}\}| = 2$. Implies, the vertices $q_{2\ell'}$ and $q_{2\ell'+1}$ are not satisfying the second condition, which is a contradiction to our assumption that L is an LDS of G'.

Case b. ℓ' **is odd:** If we proceed as in Case a, we can observe that L must contain *all* the four vertices on $(\frac{\ell'}{2}+1)$-th segment, i.e., the middle segment, (see Fig. 5.4(b)). Which is a contradiction to our assumption that no segment, representing the edge $e(v_i, v_j)$, contains more than three vertices in L.

We shall show that, by removing and/or replacing some vertices in L, a set of k vertices from N can be chosen such that the corresponding vertices in G is a vertex cover. The vertices in S account for $3n$ vertices in L (due to Claim (i)). Update L by removing the vertices in S, i.e., $L = L \setminus S$. Let $D = \{v_i \in V \mid p_i \in L \cap N\}$. If any edge $e(v_i, v_j)$ in G has none of its end vertices in D, then we do the following: consider the sequence of segments representing the edge $e(v_i, v_j)$ in the embedding. Since, both p_i and p_j are not in L, there must exist a segment having all its vertices in L (due to Claim (iii)). Consider the segment having its four vertices in L. Delete any one of the vertices on the segment and introduce p_i (or p_j). Update D and repeat the process till every edge has at least one of its end vertices in D. Note that D is a vertex cover in G with cardinality at most k (due to Claim (ii)).

Therefore, LDS-UDG is NP-complete. □

5.3 Approximation Algorithms

5.3.1 A $\frac{63}{2}$-factor approximation algorithm

Let $\mathcal{R}$ be the rectangular region containing the point set $\mathcal{P}$. The objective of this section is to find a liar's dominating set $\mathcal{P}'(\subseteq \mathcal{P})$ such that $|\mathcal{P}'| \leq \frac{63}{2}|OPT|$, where OPT is an

optimum solution. We partition the region $\mathcal{R}$ into regular hexagonal cells with side length $\frac{1}{2}$ (see Figure 3.1(a) in Chapter 3). The pseudo code to find a liar's dominating set for the points in $\mathcal{P}$ is given in Algorithm 5.1.

Algorithm 5.1 Liar's_dominating_set ($\mathcal{P}$)

Require: The set $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ of n points.
Ensure: A liar's dominating set $\mathcal{P}'$ of $\mathcal{P}$

1: Let $\mathcal{R}$ be the minimum size axis parallel rectangle containing points in $\mathcal{P}$
2: $\mathcal{P}' = \emptyset$
3: Partition the region $\mathcal{R}$ into regular hexagonal cells of side length $\frac{1}{2}$.
4: **for** (each non-empty cell H_i in the partition) **do**
5: $n_i \leftarrow$ the number of points in H_i
6: **if** $(n_i \geq 3)$ **then**
7: Choose any three arbitrary points p_i, p_j, and p_k from H_i.
8: $\mathcal{P}' = \mathcal{P}' \cup \{p_i, p_j, p_k\}$.
9: **else**
10: **if** $(n_i = 2)$ **then**
11: Let p_i and p_j are the points in H_i.
12: $\mathcal{P}' = \mathcal{P}' \cup \{p_i, p_j, p_k\}$, where $p_k \in N[p_i] \cup N[p_j]$ and is different from p_i and p_j.
13: **else**$(n_i = 1)$
14: Let p_i be the point in H_i.
15: $\mathcal{P}' = \mathcal{P}' \cup \{p_i, p_j, p_k\}$, where p_j and p_k are two neighbors of p_i, if exists, other wise, p_j is a neighbor of p_i and p_k is a neighbor of $p_j (i \neq k)$.
16: **end if**
17: **end if**
18: **end for**
19: Return $\mathcal{P}'$.

Lemma 5.3.1. *$\mathcal{P}'$ returned by Algorithm 5.1 is a liar's dominating set of $\mathcal{P}$.*

Proof. Let H_i be a non-empty cell in the hexagonal partition of $\mathcal{R}$ and $p_i \in H_i \cap \mathcal{P}$. If $|H_i \cap \mathcal{P}| \geq 2$, then the algorithm chooses at least two points from $H_i \cap \mathcal{P}$ as part of $\mathcal{P}'$ (see line numbers 8 and 12 in Algorithm 5.1). Therefore, $|N[p_i] \cap \mathcal{P}'| \geq 2$ by Observation 3.3.1. Now, if $|H_i \cap \mathcal{P}| = 1$, then $\mathcal{P}'$ contains p_i and one more point of $\mathcal{P}$, which is a neighbor of p_i (see line number 15 in Algorithm 5.1). So, in this case also $|N[p_i] \cap \mathcal{P}'| \geq 2$. Therefore, $|N[p_i] \cap \mathcal{P}'| \geq 2$ for any $p_i \in \mathcal{P}$.

Now we prove that every pair of points p_i and $p_j (p_i \neq p_j)$ in $\mathcal{P}$ have at least three points of $\mathcal{P}'$ in their closed neighborhood union.

Case 1. p_i and p_j belong to the same cell, say H_i.

1. H_i contains more than two points of $\mathcal{P}$: $\mathcal{P}'$ contains three points from H_i (see line number 8 in Algorithm 5.1). p_i and p_j may or may not be part of these three points. In either case $|(N[p_i] \cup N[p_j]) \cap \mathcal{P}'| \geq 3$ holds by Observation 3.3.1.
2. H_i contains only two points of $\mathcal{P}$: in this case both p_i and p_j must be in $\mathcal{P}'$ and one of its neighbors also must be in $\mathcal{P}'$ (see line number 12 in Algorithm 5.1). Hence, $|(N[p_i] \cup N[p_j]) \cap \mathcal{P}'| \geq 3$.

Case 2. p_i and p_j belong to distinct cells, say H_i and H_j, respectively.

1. $|(H_i \cup H_j) \cap \mathcal{P}| \geq 3$ i.e., either $|H_i \cap \mathcal{P}| \geq 2$ and $|H_j \cap \mathcal{P}| \geq 1$, or $|H_i \cap \mathcal{P}| \geq 1$ and $|H_j \cap \mathcal{P}| \geq 2$: $\mathcal{P}'$ contains at least 3 points from $(H_i \cup H_j) \cap \mathcal{P}$ (see line numbers 8, 12, and 15 in Algorithm 5.1). Therefore, $|(N[p_i] \cup N[p_j]) \cap \mathcal{P}'| \geq 3$ (by Observation 3.3.1).
2. $|(H_i \cup H_j) \cap \mathcal{P}| = 2$ i.e., $|H_i \cap \mathcal{P}| = 1$ and $|H_j \cap \mathcal{P}| = 1$: in this case, the algorithm chooses p_i and p_j as members of $\mathcal{P}'$ along with at least one neighbor of p_i (resp. p_j) (see line number 15 in Algorithm 5.1). Therefore, in this case also $|(N[p_i] \cup N[p_j]) \cap \mathcal{P}'| \geq 3$.

Thus the lemma. □

Theorem 5.3.2. *Algorithm 5.1 produces a $\frac{63}{2}$-factor approximation result for the GMLDS problem in $O(n \log n)$ time.*

Proof. Let OPT be an optimum solution for the point set $\mathcal{P}$. Consider a non-empty cell H_i in the hexagonal partition. The points in H_i can be covered by the points in H_i and also by the points in at most 18 surrounding cells (see Figure 5.5). Let $p_i \in H_i$. Observe

that the number of points in OPT dominating (covering) the point p_i is at least 2 (due to first condition of liar's domination). Hence, the points in OPT dominating the point p_i, say p_j and p_k, should be from H_i and/or from its 18 surrounding cells (shown as solid cells in Figure 5.5), and these 18 cells are the cells that are at most unit distance away from H_i.

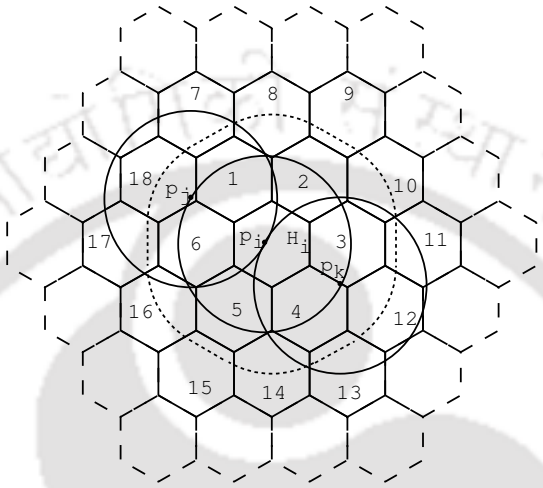


Figure 5.5: Cells within unit distance (solid cells) from H_i.

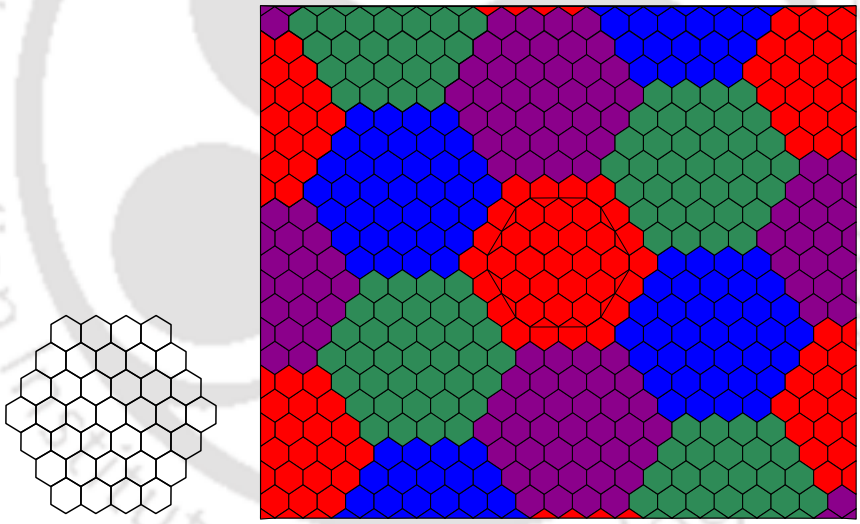
The point p_j (resp. p_k) can cover points in at most 12 cells including H_i (by Observation 4.5.4). Hence, the points covered by p_j (resp. p_k) must lie within the unit radius disk centered at p_j (resp. p_k). However they might have some common cell covered. The points p_k and p_j cover more cells when they have less number of common cells covered. This scenario happens when p_j and p_k are the end points of the diameter of the unit radius disk centered at p_i. In this case they can have at most three common cells, and cover points in 21 distinct cells. Algorithm 5.1 chooses at most 3 points for each cell. Therefore, the approximation factor is $\frac{21\times 3}{2} = \frac{63}{2}$.

For a point $p_i \in \mathcal{P}$, computing its cell number can be done in constant time as we know the coordinates of the points. We can store the non-empty grid cells in a data structure such as balanced binary tree. For a point $p_i \in \mathcal{P}$, we compute its cell number and check for the presence of the cell in the data structure. If the cell is not present, we

insert the cell in the data structure. Hence, we have in hand that how many and which points of $\mathcal{P}$ are in a cell. After processing the points we just need go through the data structure to report the required points. For one point we spend $O(\log n)$ time, hence for n points it takes $O(n \log n)$ time. Thus the running time for Algorithm 5.1 follows. □

5.3.2 Improving the approximation factor

A 37-hexagon is a combination of 37 adjacent regular hexagonal cells such that one cell is surrounded by 36 other cells (see Figure 5.6(a)). Let us consider the partition of the rectangular region $\mathcal{R}$ into 37-hexagons such that no point of $\mathcal{P}$ lies on the boundary of any 37-hexagon in the partition, and a 4 coloring scheme of it (see Figure 5.6(b)). Consider a 37-hexagon colored with A. Its adjacent six 37-hexagons are colored B, C, and D, such that opposite pair of 37-hexagons receive the same color.



(a) (b)

Figure 5.6: (a) A single 37-hexagon, and (b) a four coloring scheme of the hexagonal grid.

Lemma 5.3.3. *The minimum distance between any two same colored 37-hexagons is greater than or equal to 5.*

Consider a 37-hexagon H. H can be viewed as a 19-hexagon, say H', surrounded by 18 cells. Let us consider the convex hull overlay, say CH, of the set of corners of H' (shown as loop in Figure 5.6(b)). Observe that the maximum distance between any two points in the convex hull overlay CH is at most $\frac{5\sqrt{3}}{2}(> 4)$. Let $S = CH \cap \mathcal{P}$, $S' = \{p \in \mathcal{P} \mid \delta(p,q) \leq 1 \; for \; q \in S\}$, and $S'' = \{p \in \mathcal{P} \mid \delta(p,q) \leq 1 \; for \; q \in S'\}$, where $\delta(p,q)$ denotes the Euclidean distance between p and q. We first propose an approximation algorithm to find a liar's dominating set $S_H \subseteq S''$ for S'. Next, we use the above approximation result to design an approximation algorithm to find out a liar's dominating set for $\mathcal{P}$. Let $OPT_{S'_H}$ denotes an optimal liar's dominating set of S'.

Lemma 5.3.4. $|OPT_{S'_H}| \leq 183$.

Proof. The points in $S' \setminus S$ lie in at most 24 cells around H by definition of S' (i.e., one layer around H). Hence, the points in S' can span over 61 cells. If we choose at most three points for each cell, we get a liar's dominating set. Thus, the cardinality of $OPT_{S'_H}$ cannot be more than 183. □

Lemma 5.3.5. *If H_1 and H_2 are two same colored 37-hexagons, then $OPT_{S'_{H_1}}$ and $OPT_{S'_{H_2}}$ are independent, i.e., $OPT_{S'_{H_1}} \cap OPT_{S'_{H_2}} = \emptyset$.*

Proof. The proof follows from Lemma 5.3.3. □

The detailed pseudo code to find a liar's dominating set for the points lying in a given 37-hexagon H is given in Algorithm 5.2. For a given k ($3 \leq k \leq 183$), we choose all possible $t = 1, 2, \ldots, k-1$ combinations of points in S'' to find a 2-tuple dominating set of S'. For each combination, we check the combination of points is a 2-tuple dominating set or not (Line number 5). While considering the subsets, if there exists a subset S_H that is a 2-tuple dominating set, then for every distinct pair of points p_i and p_j in S', we check whether $|(N[p_i] \cup N[p_j]) \cap S_H| \geq 3$ or not (Line number 7). A subset S_H satisfying the above condition is reported and is a liar's dominating set for S'. In the algorithm Boolean variable $flag$ is used to ensure that the set returned by the algorithm is a feasible solution.

Lemma 5.3.6. *For a given 37-hexagon H, Algorithm 5.2 produces a solution S_H for the set S' from S'' with size is at most $\frac{183}{k} \times |OPT_H|$, where $3 \leq k \leq 183$ and OPT_H is an optimum solution for the points lying in H.*

Proof. If the algorithm cannot produce a solution of size $k-1$ for given k ($3 \leq k \leq 183$), then $|OPT_H| \geq k$. Observe that, our algorithm may produce a solution S_H whose size is 183, in the worst. Hence $\frac{|S_H|}{|OPT_H|} \leq \frac{183}{k}$. □

Algorithm 5.2 Liar's_dominating_set (H, k)

Require: Point set $\mathcal{P}$, a 37-hexagon H and an integer $3 \leq k \leq 183$.

Ensure: A liar's dominating set of size $\leq k-1$ for the set S' from S'' (if exists), otherwise a set $S_H (\subseteq S'')$ of size at most 183.

1: Obtain sets S, S', and S''.
2: $flag = 0$.
3: **for** $(t = 1, 2, \ldots, k-1)$ **do**
4: **for** (each combination S_H of t points in S'') **do**
5: **if** $(|S_H \cap N[p_i]| \geq 2 \ \forall p_i \in S')$ **then**
6: **for** (every distinct pair of points p_i and p_j in S') **do**
7: **if** $(|(N[p_i] \cup N[p_j]) \cap S_H| \geq 3)$ **then**
8: $flag = 1$.
9: **else**
10: $flag = 0$.
11: **break**/*break the for loop in line number 6 */
12: **end if**
13: **end for**
14: **end if**
15: **if** $(flag = 1)$ **then**
16: Return S_H.
17: **break**/*break the for loop in line number 3 */
18: **end if**
19: **end for**
20: **end for**
21: $S_H \leftarrow \emptyset$
22: **if** $(flag = 0)$ **then**
23: Consider any three points from each non-empty cell corresponding to S' and add it to S_H (if a non-empty cell contains less than three points we choose the remaining points from its neighbors as in Algorithm 5.1).
24: **end if**
25: Return S_H.

Lemma 5.3.7. *Algorithm 5.2 runs in $O(n^{k+1}\Delta)$ time, where $\Delta = \max\{|N[p]| : p \in \mathcal{P}\}$ and $3 \leq k \leq 183$.*

Proof. Algorithm 5.2 chooses all possible $k-1$ combinations for a given k. If any of these combinations satisfies liar's domination conditions, Algorithm 5.2 reports the combination of points. If it is not possible to find a solution of size $k-1$, the algorithm chooses at most three points for each non-empty cell (like in Algorithm 5.1). Steps 4-19 in Algorithm 5.2 can be done in $O(n^{t-1})(O(\Delta) + O(n^2\Delta)) = O(n^{t+1}\Delta)$. Hence, steps 3-20 take $\sum_{t=1}^{k-1} O(n^{t+1})\Delta$ time. Steps 1 and 22 can be done in $O(n \log n)$ time. Therefore the total running time of Algorithm 5.2 is $O(n^{k+1}\Delta)$. Thus the running time result follows. □

We consider each 37-hexagon and compute a feasible solution (using Algorithm 5.2) for the points lying in it. Two same colored 37-hexagons can be solved independently as the minimum distance between them is greater than 4. Let S_j be the union of solutions generated by the algorithm for the 37-hexagons colored j, for $j \in \{A, B, C, D\}$. The set $\mathcal{S} = \bigcup_{j \in \{A,B,C,D\}} S_j$ is reported.

Theorem 5.3.8. *$\mathcal{S}$ is a liar's dominating set of $\mathcal{P}$*

Proof. In Algorithm 5.2, for a 37-hexagon H, we find a liar's dominating set S_H for S' from S'' (refer the definitions of S' and S'' defined previously in this section). Now, $\mathcal{S} = \bigcup_{H \in \{\text{all 37-hexagons in } \mathcal{R}\}} S_H$. Thus, the theorem follows.

□

Theorem 5.3.9. *The 4-coloring scheme gives a $\frac{732}{k}$-factor approximation algorithm for the geometric liar's domination problem in the plane and runs in $O(n^{k+1}\Delta)$ time, where $3 \leq k \leq 183$.*

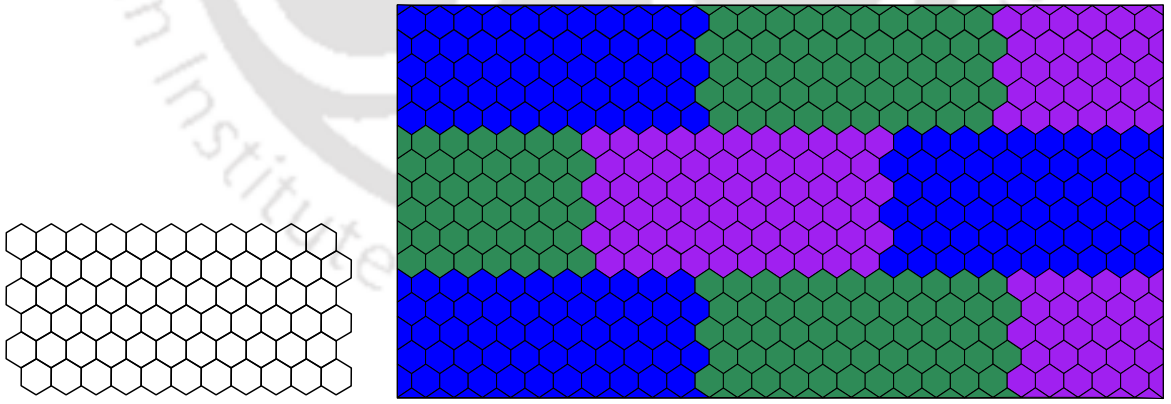
Proof. By Lemma 5.3.3, any two same colored 37-hexagons are greater than five units apart. Therefore, we can solve them independently. Let OPT_j be the union of solutions in optimal solution for the 37-hexagons colored j, for $j \in \{A, B, C, D\}$. Also, let OPT

be an optimum solution for the point set $\mathcal{P}$, hence, $|OPT_j| \leq |OPT|$ for each $j \in \{A, B, C, D\}$ and $OPT = OPT_A \cup OPT_B \cup OPT_C \cup OPT_D$. Observe that, OPT_j is the optimum for color class j, where we dominate the sets S' with respect to the group of 37-hexagons of color j using the points from S'' and not just S'. The 4-coloring scheme reports the set $\mathcal{S}$, which is the union of the solutions for all 37-hexagons. Therefore, $|\mathcal{S}| \leq \frac{183}{k}(|OPT_A| + |OPT_B| + |OPT_C| + |OPT_D|)$. Implies, $|\mathcal{S}| \leq \frac{732}{k} \times |OPT|$ as $|OPT_i| \leq |OPT|$ for each $i \in \{A, B, C, D\}$.

The running time result follows from Lemma 5.3.7 and the fact that a point in $\mathcal{P}$ participates a finite number of times in the algorithm. Hence the theorem. □

5.3.3 Further improvement of the approximation factor

The best approximation factor that can be achieved using the algorithm proposed in Subsection 5.3.2 is 4 for $k = 183$. We can extend the idea used for 37-hexagonal partition further to get best approximation factor 3. A 66-hexagon is a combination of 66 cells arranged in six rows and each row contains 11 cells (see Figure 5.7(a)). We partition the rectangular region $\mathcal{R}$ containing the points in $\mathcal{P}$ into 66-hexagons such that no point of $\mathcal{P}$ lies on the boundary and consider a 3-coloring scheme of it (see Figure 5.7(b)).



(a) (b)

Figure 5.7: (a) A single 66-hexagon, and (b) a three coloring scheme of the hexagonal grid.

By using the technique in Subsection 5.3.2, we can get $\frac{282}{k}$-factor approximation algorithm for the points lying in a 66-hexagon, where $3 \leq k \leq 282$, and we have the following theorem.

Theorem 5.3.10. *The 3-coloring scheme gives a $\frac{846}{k}$-factor approximation algorithm for the geometric liar's domination problem in the plane and the algorithm runs in $O(n^{k+1}\Delta)$ time, where $3 \leq k \leq 282$.*

5.4 A PTAS for the GMLDS problem

For a given UDG, $G = (V, E)$, and a parameter $\varepsilon > 0$, we propose an algorithm which produces a liar's dominating set of cardinality no more than $(1 + \varepsilon)$ times the size of a minimum liar's dominating set in G.

The proposed PTAS is based on the concept of m-separated collection of subsets of V $(m \geq 4)$. A collection $S = \{S_1, S_2, \ldots, S_k\}$ such that $S_i \subseteq V$ for $i = 1, 2, \ldots, k$, is said to be an m-separated collection, if $d_G(S_i, S_j) > m$, for $1 \leq i, j \leq k$ (see Figure 5.8 for a 4-separated collection). Nieberg and Hurink [77] considered 2-separated collection to propose a PTAS for the minimum dominating set problem on unit disk graphs.

Lemma 5.4.1. *Let $S = \{S_1, S_2, \ldots, S_k\}$ be an m-separated collection. If $|S_i| \geq 3$ for $1 \leq i \leq k$, and $m \geq 4$, then $\sum_{i=1}^{k} |LD_{opt}(S_i)| \leq |LD_{opt}(V)|$.*

Proof. Recall that for a set $A \subseteq V$, $N_G[A] = \bigcup_{v \in A} N_G[v]$. For $i \neq j$, $1 \leq i, j \leq k$, $N_G[S_i] \cap N_G[S_j] = \emptyset$ and also $LD_{opt}(S_i) \cap LD_{opt}(S_j) = \emptyset$ as S_i and S_j are m-separated $(m \geq 4)$ i.e., $d_G(S_i, S_j) > 4$. Let $S_i' = \{u \in V \mid v \in S_i \text{ and } d_G(u, v) \leq 2\}$ for $i = 1, 2, \ldots, k$. Observe that $S_i \subseteq S_i'$ and $S_i' \cap LD_{opt}(V)$ is a liar's dominating set of S_i for $i = 1, 2, \ldots, k$. Since $d_G(S_i, S_j) > 4$ for $i \neq j, 1 \leq i, j \leq k, S_i' \cap S_j' = \emptyset$. Therefore, $(S_i' \cap LD_{opt}(V)) \cap (S_j' \cap LD_{opt}(V)) = \emptyset$ and $\bigcup_{i=1}^{k} (s_i' \cap LD_{opt}(V)) \subseteq LD_{opt}(V)$. Also, $LD_{opt}(S_i) \subseteq S_i' \cap LD_{opt}(V)$ for $i = 1, 2, \ldots, k$, $S_i' \cap LD_{opt}(V)$ is a liar's dominating set of S_i, and $LD_{opt}(S_i)$ is a minimum size liar's dominating set of S_i. Thus, $\bigcup_{i=1}^{k} LD_{opt}(S_i) \subseteq (S_i' \cap LD_{opt}(V)) \subseteq LD_{opt}(V)$. Hence, the result of the lemma follows. □

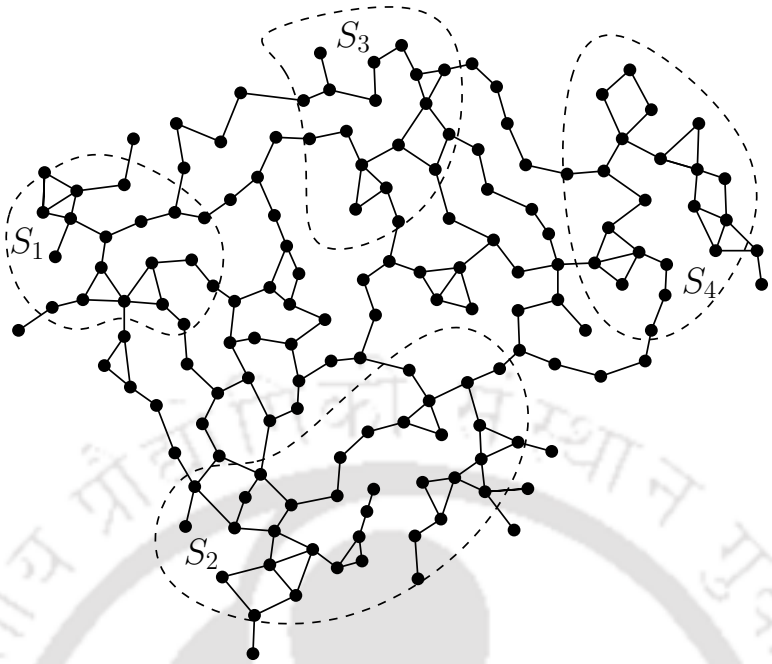


Figure 5.8: An example of a 4-separated collection $S = \{S_1, S_2, S_3, S_4\}$.

Lemma 5.4.2. *Let $S = \{S_1, S_2, \ldots, S_k\}$ be an m-separated collection ($m \geq 4$) in $G = (V, E)$, and $N_1, N_2, \ldots, N_k$ be the subsets of V with $S_i \subseteq N_i$ for all $i = 1, 2, \ldots, k$. If there exists a $\rho \geq 1$ such that $|LD_{opt}(N_i)| \leq \rho |LD_{opt}(S_i)|$ holds for all $i = 1, 2, \ldots, k$, and if $\bigcup_{i=1}^{k} LD_{opt}(N_i)$ forms a liar's dominating set in G, then the value of $\sum_{i=1}^{k} |LD_{opt}(N_i)|$ is at most ρ times the size of a minimum liar's dominating set in G.*

Proof. $|\bigcup_{i=1}^{k} LD_{opt}(N_i)| = \sum_{i=1}^{k} |LD_{opt}(N_i)| \leq \rho \sum_{i=1}^{k} |LD_{opt}(S_i)| \leq \rho |LD_{opt}(V)|$. The latter inequality follows from Lemma 5.4.1. □

In the proposed PTAS we have chosen the concept of 4-separated collection, to make $LD_{opt}(N_i) \cap LD_{opt}(N_j) = \emptyset$, for $i \neq j$.

5.4.1 Algorithm

In this section, we discuss the construction of a 4-separated collection $S = \{S_1, S_2, \ldots, S_k\}$ and subsets $N_1, N_2, \ldots, N_k$ of V such that $S_i \subseteq N_i$ for all $i = 1, 2, \ldots, k$. The al-

gorithm proceeds in an iterative manner. Initially $V_1 = V$. In i-th iteration the algorithm computes S_i and N_i. For a given $\varepsilon > 0$, i-th iteration of the algorithm starts with an arbitrary vertex $v \in V_i$ and increase the value of $r(= 1, 2, \ldots)$ as long as $|LD(N^{r+4}[v])| > \rho|LD(N^r[v])|$ holds. Here, $LD(N^{r+4}[v])$ and $LD(N^r[v])$ are liar's dominating sets of $N^{r+4}[v]$ and $N^r[v]$, respectively, and $\rho = 1 + \varepsilon$. The smallest r violating the above condition, say $\hat{r}$, is obtained. Set $S_i = N^{\hat{r}}[v]$ and $U_i = N^{\hat{r}+4}[v]$. Now removal of U_i from V_i may lead to the following two cases: (i) there might be some isolated vertex/vertices in $V_i \setminus U_i$, and/or (ii) connected component(s) of size two in $V_i \setminus U_i$. In case (i), for each isolated vertex u find $x, y \in U_i$ such that $\{u, x, y\}$ forms a connected component and update U_i as follows: $U_i = U_i \setminus \{x, y\}$. In case (ii), for each such pair of vertices u, w find $x \in U_i$ such that $\{u, w, x\}$ forms a connected component and update U_i as follows: $U_i = U_i \setminus \{x\}$. Set $N_i = U_i$ and $V_{i+1} = V_i \setminus N_i$. The process stops if $V_{i+1} = \emptyset$ and returns the sets S_i and N_i. The collection of the sets S_i is a 4-separated collection. The pseudo code is given in Algorithm 5.3.

Liar's dominating set of the r-th neighborhood of a vertex v, $LD(N^r[v])$, can be computed with respect to G as follows. We successively find maximal independent set I_1, I_2 and I_3 such that $I_1 \cap I_2 \cap I_3 = \emptyset$. Now $I_1 \cup I_2 \cup I_3$ is a liar's dominating set for $N^r[v] \setminus I_1$ as every vertex not in I_1 is either belongs to $I_2 \cup I_3$ or adjacent to at least one vertex in each I_1, I_2 and I_3. To ensure the liar's domination conditions for the vertices in I_1, for each vertex u in I_1 we add two arbitrary vertices from the neighborhood of u, if exist. If u has only one neighbor, say u', then we add u' and one of its neighbor in the solution. The pseudo code is given in Algorithm 5.4.

In summary, Algorithm 5.4 deals with obtaining a liar's dominating set in r-th neighborhood of a vertex. Algorithm 5.3 deals with obtaining a 4-separated collection $S = \{S_1, S_2, \ldots, S_k\}$ and collection $N = \{N_1, N_2, \ldots, N_k\}$ in G such that $S_i \subseteq N_i \subseteq V$ and using Algorithm 5.4 as a sub-routine it computes a liar's dominating set for G.

Lemma 5.4.3. *$LD(N^r[v])$ returned by Algorithm 5.4 is a liar's dominating set of $N^r[v]$.*

Proof. Let $u \in V$ be any vertex in G.

Algorithm 5.3 Liar's_dominating_set (V)

Require: An undirected graph $G = (V, E)$ with $|V| \geq 3$ and an arbitrary small real number $\varepsilon > 0$

Ensure: A liar's dominating set $\mathcal{D}$ of V

1: $i = 0$ and $V_{i+1} = V$
2: $\mathcal{D} = \emptyset$
3: $\rho = 1 + \varepsilon$
4: **while** $(V_{i+1} \neq \emptyset)$ **do**
5: pick an arbitrary $v \in V_{i+1}$
6: $N^0[v] = v$
7: $r = 1$
8: **while** $|(LD(N^{r+4}[v])| > \rho |LD(N^r[v])|$ **do** ▷ call Algorithm 5.4
9: $r = r + 1$
10: **end while**
11: $\hat{r} = r$ ▷ the smallest r violating while condition in step 8
12: $i = i + 1$ ▷ the index i keeps track of the number of iterations
13: $S_i = N^{\hat{r}}[v]$
14: $U_i = N^{\hat{r}+4}[v]$
15: **if** $(V_{i+1} \setminus N^{\hat{r}+4}[v]$ contains isolated components of size 1 and/or 2) **then**
16: **for** (each component, $\{u\}$, of size 1) **do**
17: find $x, y \in U_i^4$ such that $\{u, x, y\}$ is a connected component
18: $U_i^4 = U_i^4 \setminus \{x, y\}$
19: **end for**
20: **for** (each component, $\{u, w\}$, of size 2) **do**
21: find $x \in U_i^4$ such that $\{u, w, x\}$ is a connected component
22: $U_i^4 = U_i^4 \setminus \{x\}$
23: **end for**
24: **end if**
25: $N_i = U_i^4$
26: $\mathcal{D} = \mathcal{D} \cup LD(N_i)$ ▷ call Algorithm 5.4
27: $V_{i+1} = V_i \setminus N_i$
28: **end while**
29: **return** $\mathcal{D}$

We prove the first condition of liar's domination in the following two cases.

Case 1. $u \in N^r[v] \setminus (I_1 \cup I_2 \cup I_3)$

Observe that $I_1 \cap I_2 \cap I_3 = \emptyset$. The vertex u is dominated by at least three vertices as in each round u is dominated by at least one vertex (see **for** loop in line number 4 in Algorithm 5.4). Thus, $|N_G[u] \cap LD(N^r[v])| \geq 3$, satisfies the first condition.

Algorithm 5.4 Liar's_dominating_set($N^r[v]$)

Require: The r-th neighborhood of a vertex v i.e., $N^r[v]$
Ensure: A liar's dominating set $LD(N^r[v])$ of $N^r[v]$

1: $j = 0$ and $I_j = \emptyset$
2: $LD(N^r[v]) = \emptyset$
3: $N = N^r[v]$
4: **for** ($j = 1$ *to* 3) **do**
5: **if** ($N \neq \emptyset$) **then**
6: $I_j = MIS(N \setminus I_{j-1})$
7: $LD(N^r[v]) = LD(N^r[v]) \cup I_j$
8: $N = N \setminus I_j$
9: **end if**
10: **end for**
11: **for** every $u \in I_1$ **do**
12: **if** $|N_G[u]| > 2$ **then**
13: add two arbitrary vertices from $N_G[u]$ to $LD(N^r[v])$
14: **else**
15: add w and its neighbor to $LD(N^r[v])$ ▷ w is the neighbor of u
16: **end if**
17: **end for**
18: **return** $LD(N^r[v])$

Case 2. $u \in I_1 \cup I_2 \cup I_3$

We consider the following two sub cases.

a. $u \in I_2 \cup I_3$

In this case $|N_G[u] \cap LD(N^r[v])| \geq 2$ holds as every vertex in $I_2 \cup I_3$ has at least one neighbor in I_1.

b. $u \in I_1$

For every vertex u in I_1, $LD(N^r[v])$ contains a neighbor of u or u's neighbor's neighbor (see line number 11 in Algorithm 5.4). Hence, $|N_G[u] \cap LD(N^r[v])| \geq 2$ holds.

Now we prove the second condition of liar's domination. Let u and w are any two vertices in $N^r[v]$. We prove the inequality $|(N_G[u] \cup N_G[w]) \cap LD(N^r[v])| \geq 3$ by considering the following cases.

Case 1. $u, w \in N^r[v] \setminus (I_1 \cup I_2 \cup I_3)$

Observe that both u and w have a neighbor in each I_1, I_2 and I_3. Hence $|(N_G[u] \cup N_G[w]) \cap LD(N^r[v])| \geq 3$.

Case 2. $u \in N^r[v] \setminus (I_1 \cup I_2 \cup I_3)$ and $w \in I_1 \cup I_2 \cup I_3$ (proof of the other case is similar)

In this case the inequality $|(N_G[u] \cup N_G[w]) \cap LD(N^r[v])| \geq 3$ is true as $|N_G[u] \cap LD(N^r[v])| \geq 3$ and $|N_G[w] \cap LD(N^r[v])| \geq 2$.

Case 3. $u, w \in I_1 \cup I_2 \cup I_3$

The inequality $|(N_G[u] \cup N_G[w]) \cap LD(N^r[v])| \geq 3$ holds in this case as $u, w \in LD(N^r[v])$ and each have a neighbor (different from w, u, respectively) in $LD(N^r[v])$.

Thus the lemma follows. □

Lemma 5.4.4. $|LD(N^r[v])| \leq O(r^2)$.

Proof. Algorithm 5.4 computes $LD(N^r[v])$ by first computing maximal independent sets I_1, I_2, and I_3 subsequently. After computing I_1, I_2 and I_3, the algorithm adds two vertices for each vertex in I_1 to ensure $LD(N^r[v])$ is a feasible solution. Without loss of generality we assume that $|I_1| \geq |I_2|, |I_3|$. Hence, $|LD(N^r[v])| = 3|I_1| + |I_2| + |I_3| \leq 5 \times |I_1| \leq 5 \times \frac{\pi(r+1)^2}{\pi(1)^2} = O(r^2)$. The last inequality follows from the standard area argument, the number of non-intersecting disks of unit radius can be packed in the larger disk of radius $r + 1$ centered at v. □

Lemma 5.4.5. *In each iteration of Algorithm 5.3, there exists an r violating the condition $|(LD(N^{r+4}[v])| > \rho|LD(N^r[v])|$, where $\rho = 1 + \varepsilon$*

Proof. We prove the lemma by contradiction. Suppose there exists $\varepsilon > 0$ and $v \in V$ such that $|(LD(N^{r+4}[v])| > \rho|LD(N^r[v])|$ for $r = 1, 2, \ldots$. Observe that $|LD(N^2[v])| \geq 3$ as there exists at least three vertices in G.

If r is even, we get

$$(r+5)^2 \geq |(LD(N^{r+4}[v])| > \rho|LD(N^r[v])| > \cdots > \rho^{\frac{r}{2}}|LD(N^2[v])| \geq 3\rho^{\frac{r}{2}}.$$

If r is odd, we have

$$(r+5)^2 \geq |(LD(N^{r+4}[v])| > \rho|LD(N^r[v])| > \cdots > \rho^{\frac{r-1}{2}}|LD(N^3[v])| \geq 3\rho^{\frac{r-1}{2}}.$$

In both the cases the first inequality follows from Lemma 5.4.4. Hence,

$$(r+5) > \begin{cases} (\sqrt{\rho})^r, & \text{if } r \text{ is even} \\ (\sqrt{\rho})^{r-1}, & \text{if } r \text{ is odd} \end{cases} \tag{5.1}$$

Note that in the inequality (5.1), the right hand part is a function having exponential in r and the left hand part is a polynomial in r. For sufficiently large r both the inequalities cannot be true in either case. Hence we arrived at contradiction. Thus there exist an r violating the condition. □

Lemma 5.4.6. *The smallest r violating the inequality (5.1) is a constant and is bounded by $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$.*

Proof. Lemma 5.4.5 suggests that the smallest r violating the condition cannot be sufficiently large but a constant. Let $\hat{r}$ is the smallest r violating the condition i.e., when $r = \hat{r}$ the inequalities in (5.1) are violated. Using the inequality $\log(1+\varepsilon) > \frac{\varepsilon}{2}$ for $0 < \varepsilon < 1$, we have $\hat{r} \leq O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ as follows.

Consider the inequality $(1+\varepsilon)^x < x^2$ for a fixed $\varepsilon > 0$. We prove that there exist ε' such that $(1+\varepsilon)^x < x^2 < (x+5)^2$ for $0 < \varepsilon < \varepsilon'$. The latter inequality is trivial. Let $x = \frac{4}{\varepsilon}\log\frac{1}{\varepsilon}$. By taking the logarithm on both sides of the former inequality, we get, $\log 4 + \log\log\frac{1}{\varepsilon} > 0$. Note that, we can always find an ε' such that $\log 4 + \log\log\frac{1}{\varepsilon} > 0$ for $0 < \varepsilon < \varepsilon'$. Therefore, $(1+\varepsilon)^x < x^2$ holds for sufficiently smaller ε values and hence, $\hat{r} \leq \frac{4}{\varepsilon}\log\frac{1}{\varepsilon}$. □

Lemma 5.4.7. *For a given $v \in V$, liar's dominating set $LD_{opt}(N_i)$ of N_i can be computed in polynomial-time.*

Proof. Note that $N_i \subseteq N^{r+4}[v]$. The size of a liar's dominating set $LD(N_i)$ of N_i obtained by Algorithm 5.4 is bounded by $O(r^2)$ (by Lemma 5.4.4). Again, $r = O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ by Lemma 5.4.6. Therefore, the size of the minimum size liar's dominating set $LD_{opt}(N_i)$

of N_i is bounded by a constant. The process of checking whether a given set is a liar's dominating set or not can be done in polynomial-time. Therefore, we can consider every subset of N_i as a possible liar's dominating set and check whether it is a liar's dominating set or not in polynomial-time. Finally, the minimum size liar's dominating set is reported. Thus the lemma follows. □

Lemma 5.4.8. *For the collection of neighborhoods* $\{N_1, N_2, \ldots, N_k\}$ *created by Algorithm 5.3, the union* $\mathcal{D} = \bigcup_{i=1}^{k} LD(N_i)$ *is a liar's dominating set in* G.

Proof. We first prove that for every $v \in V, |N_G[v] \cap \mathcal{D}| \geq 2$. Observe that $\bigcup_{i=1}^{k} N_i = V$ as $V_{i+1} = V_i \setminus N_i$ and $N_i \subseteq V_i$. Thus, every vertex $v \in N_i$ for some $1 \leq i \leq k$. By Lemma 5.4.3, $|N_G[v] \cap \mathcal{D}| \geq 2$ is satisfied.

Now we prove the second condition. Consider two arbitrary vertices $u, v \in V$. The following cases may arise.

Case 1. $u, v \in N_i$ for some $1 \leq i \leq k$

Since $LD(N_i)$ is the liar's dominating set of N_i in G, we have, $|(N_G[u] \cup N_G[v]) \cap LD(N_i)| \geq 3$ for every $u, v \in N_i$. Hence, $|(N_G[u] \cup N_G[v]) \cap \mathcal{D}| \geq 3$ for every $u, v \in V$.

Case 2. $u \in N_i$ and $v \in N_j$ for some $i \neq j$ and $1 \leq i, j \leq k$

If u and v are not adjacent in G, the proof is trivial. Hence, we assume that $(u, v) \in E$ i.e., u and v are adjacent in G. Now the following sub cases may arise.

a. $u \in LD(N_i)$ and $v \in LD(N_j)$

Observe that $|N_G[u] \cap LD(N_i)| \geq 2$ and $|N_G[v] \cap LD(N_j)| \geq 2$ as $LD(N_i)$ and $LD(N_j)$ are liar's dominating sets of N_i and N_j, respectively. Hence, u has a neighbor, say w, in $LD(N_i)$, similarly v has also a neighbor, say x, in $LD(N_j)$. However, may be $w = x$ or may not be. In either case $|(N_G[u] \cup N_G[v]) \cap \mathcal{D}| \geq 3$ holds.

b. $u \notin LD(N_i)$ and $v \in LD(N_j)$ (the other case proof is similar)

Since $LD(N_i)$ is a liar's dominating set of N_i, we have $|N_G[u] \cap LD(N_i)| \geq 2$. Hence, $|(N_G[u] \cup N_G[v]) \cap \mathcal{D}| \geq 3$ is true as v is part of the solution.

c. $u \notin LD(N_i)$ and $v \notin LD(N_j)$

The proof is similar to the previous cases.

Thus the lemma is proved. □

Corollary 5.4.9. *For the collection $N = \{N_1, N_2, \ldots, N_k\}$ created by Algorithm 5.3, the union $\mathcal{D}^* = \bigcup_{i=1}^{k} LD_{opt}(N_i)$ is a liar's dominating set.*

Theorem 5.4.10. *For a given UDG, $G = (V, E)$, and an $\varepsilon > 0$, we can design a $(1+\varepsilon)$-factor approximation algorithm (PTAS) to find a liar's dominating set of G with running time $n^{O(c^2)}$, where $c = O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$.*

Proof. Note that Algorithm 5.3 generates the collection of sets $S = \{S_1, S_2, \ldots, S_k\}$ and $N = \{N_1, N_2, \ldots, N_k\}$ such that S is a 4-separated collection of V with $S_i \subseteq N_i$ for each $i \in \{1, 2, \ldots, k\}$ and $\bigcup_{i=1}^{k} N_i = V$ with $N_i \cap N_j = \emptyset$ for $i \neq j$. Corollary 5.4.9 suggests that $\mathcal{D}^* = \bigcup_{i=1}^{k} LD_{opt}(N_i)$ is a liar's dominating set of G. The approximation bound follows from Lemma 5.4.1, and Lemma 5.4.2.

Let $|N_i| = n_i$ for $1 \leq i \leq k$. By Lemma 5.4.7, an optimal liar's dominating set $LD_{opt}(N_i)$ of N_i can be computed in $n_i^{O(\frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon})}$ time. Therefore, the total running time to compute $\mathcal{D}^*$ is $\sum_{i=1}^{k} n_i^{O(\frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon})} \leq n^{O(\frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon})}$. □

5.5 Conclusion

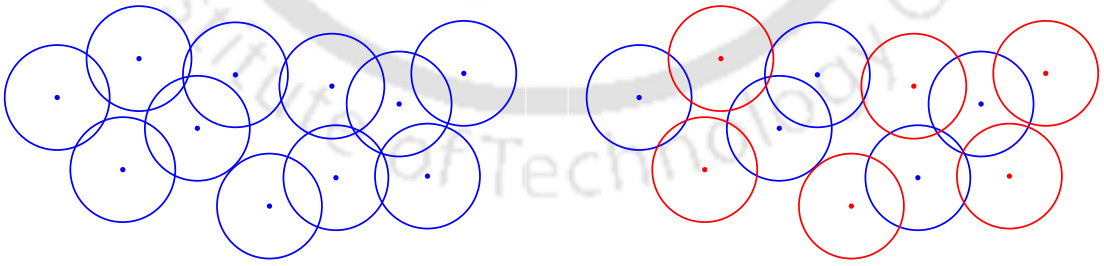
In this chapter, we considered the geometric liar's dominating set (GMLDS) problem. We proved that the GMLDS problem is NP-complete. We proposed a simple $O(n \log n)$ time $\frac{63}{2}$-factor, $\frac{732}{k}$-factor ($3 \leq k \leq 183$), and $\frac{846}{k}$-factor ($3 \leq k \leq 282$) approximation algorithms. The $\frac{63}{2}$-factor approximation algorithm runs in $O(n \log n)$ time, where as the running time of the remaining algorithms are $O(n^{k+1}\Delta)$ for their respective k. Finally, we proposed a PTAS for the GMLDS problem.

Chapter 6

Geometric Maximum (Weighted) Independent Set Problem

In this chapter, we study the geometric versions of the maximum independent set and maximum weighted independent set problems, defined as follows:

1. **Geometric maximum independent set (GMIS) problem** : *Given a set $\mathcal{P}$ of n points in $\mathbb{R}^2$, find a maximum cardinality set $\mathcal{P}' \subseteq \mathcal{P}$ such that the points in $\mathcal{P}'$ are pairwise independent i.e., the Euclidean distance between any two points in $\mathcal{P}'$ is greater than 1. In other words, if disks of radii $\frac{1}{2}$ are centered at the points in $\mathcal{P}$, find a maximum set of non-intersecting disks (see Figure 6.1).*



(a) (b)

Figure 6.1: (a) A set of disks in the plane, and (b) a maximum set of non-intersecting disks (shown in red).

2. **Geometric maximum weighted independent set (GMWIS) problem** : *Given a set $\mathcal{P}$ of n points in $\mathbb{R}^2$ and a weight function $w : \mathcal{P} \to \mathbb{R}^+$, find a set $\mathcal{P}' \subseteq \mathcal{P}$ such that the points in $\mathcal{P}'$ are pairwise independent and the sum of the weights of the points in $\mathcal{P}'$ is maximized.*

The goal of this chapter is to propose (i) a 2-factor approximation algorithm and a PTAS for the GMIS problem, and (ii) 2.16-factor and 2-factor approximation algorithms for the GMWIS problem.

6.1 Geometric Maximum Independent Set Problem

6.1.1 Preliminaries

Let $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ be a set of n points in $\mathbb{R}^2$. We use $x(p_i), y(p_i)$ to represent the x and y coordinates of the point $p_i \in \mathcal{P}$, respectively. Let $d(p_i, p_j)$ denotes the Euclidean distance between p_i and p_j. We say that two points p_i and p_j in $\mathcal{P}$ are independent (some times we say that p_i is an independent point of p_j and vice versa in the rest of the chapter) if $d(p_i, p_j) > 1$. Our objective is to find a maximum size subset $\mathcal{P}'(\subseteq \mathcal{P})$ such that all the points in $\mathcal{P}'$ are mutually independent. Without loss of generality, we assume that no two points in $\mathcal{P}$ have the same x coordinate. A *horizontal strip* H of width ℓ is the region in the plane bounded by two horizontal parallel lines and the distance between the lines is ℓ. Consider a horizontal strip H of width 1 and let $\mathcal{Q} = \{p_1, p_2, \ldots, p_m\}(\subseteq \mathcal{P})$ be the set of points lying in H in increasing order of their x-coordinates.

We define the set $S_{i,j}$ as follows: (i) all the points in $S_{i,j}$ are mutually independent, (ii) $S_{i,j}$ is a maximum cardinality subset of $\mathcal{Q}$, and (iii) p_j and p_i are the two right most points in $S_{i,j}$ with $j < i$. Let $n(S_{i,j})$ denotes the number of points in $S_{i,j}$. We use $S_i = \{S_{i,j} \mid 1 \leq j < i\}$ to denote the collections of sets $S_{i,j}$ for a fixed i. We say that two points p_u and p_v in $S_{i,j}$ are *consecutive* if $x(p_u) < x(p_v)$ and there is no other point p_w

in $S_{i,j}$ such that $x(p_u) < x(p_w) < x(p_v)$. For convenience, the set $S_{i,j}$ can be viewed as a chain $C_{i,j}$. In general, a *chain* is a series of connected line segments. In our context, the chain $C_{i,j}$ (corresponds to the set $S_{i,j}$) is defined by joining the consecutive points in $S_{i,j}$ using line segments from left to right. Therefore, S_i can be viewed as a collection of chains ending at p_i (see Figure 6.2). Note that these chains may or may not have a common point(s) except p_i.

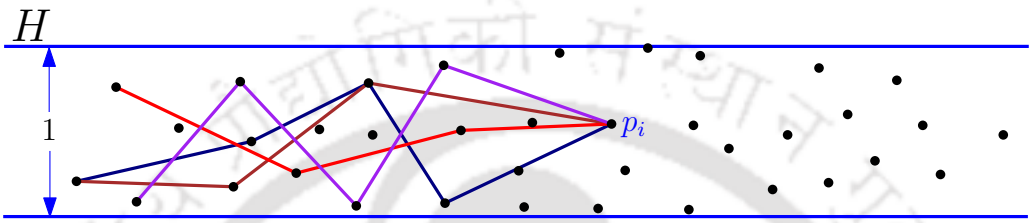


Figure 6.2: Pictorial representation of the collection S_i in the form of chains (not all are drawn).

Now, we propose an algorithm to solve the GMIS problem defined on a horizontal strip H of width 1 optimally. A detailed algorithm is given in Subsection 6.1.2. The basic steps in our algorithm to find a GMIS of the set $\mathcal{Q} = \{p_1, p_2, \ldots, p_m\}$ of points lying inside H are as follows : (i) iteratively process the points one after the other from left to right, (ii) while processing a point p_i we extend the longest chain (a chain of maximum length) ending at p_j till p_i for $1 \leq j < i$, if possible. After processing all the points in the strip H we obtain a GMIS of $\mathcal{Q}$ that corresponds to a longest chain.

Let a variable n_i be associated with each point p_i in the strip and is used to store the length of a longest chain ending at p_i, i.e., $n_i = \max\{n(s_{i,j}) \mid j < i\}$. Initially we set $n_i = 0$ for every point p_i in the strip, which indicates that the maximum length of a chain ending at p_i is zero. The value of n_i gets updated during the point p_i is being processed and reaches its final value once the i-th iteration is done. We also define the following sets: $S_i^{\alpha} = \{p_j \in \mathcal{Q} \mid p_j \in S_{i,j}$ with $j < i$ and $|S_{i,j}| = n_i - \alpha\}$ for $\alpha = 0, 1, 2, \cdots$. These sets play a major role in the proposed algorithm. We use $FPVD(S)$ to denote the farthest-point Voronoi diagram (FPVD) [11] of a point set S.

6.1.2 A 2-factor approximation algorithm

Let $\mathcal{R}$ be the axis parallel rectangular region in the plane containing the point set $\mathcal{P}$. We partition the region containing the points in $\mathcal{P}$ into disjoint strips $H_1, H_2, \ldots, H_\nu$ of width 1 using the horizontal lines at y-coordinates $h_1, h_2, \ldots, h_\nu + 1$ such that no point in $\mathcal{P}$ lies on any horizontal line. The i-th strip H_i contains the points $P_i = \{p \in \mathcal{P} \mid h_i < y(p) < h_{i+1}\}$. We compute a GMIS for each nonempty strip separately. In this section, we present a dynamic programming based algorithm with the help of FPVD to compute a GMIS of the points lying in a horizontal strip H of width 1.

Description of our algorithm to find a GMIS for a given set $\{p_1, p_2, \ldots, p_m\}$ of points lying in a strip H of width 1 is as follows: let $\mu > 1$ be a predefined sufficiently large constant. We obtain a GMIS of the points $\{p_1, p_2, \ldots, p_\mu\}$ in a naive way. We process the points from $p_{\mu+1}$ one after the other from left to right. For every point p_i in the strip we maintain a collection of sets $S_i = \{S_{i,j}\}$, where $j < i$. We compute these sets for $1 < i \leq \mu$ in brute force manner (because, we should have the sets $S_{i,j}$ in hand for every $1 < j < i \leq \mu$ before processing the point $p_{\mu+1}$) and for $i > \mu$ on the fly while p_i is being processed. We define $S_{i,j} = \emptyset$ if p_i and p_j are not independent. The points in the sets are stored in increasing order of their x-coordinate.

We also maintain the three sets S_i^0, S_i^1, S_i^2 (defined in Subsection 6.1.1) and their respective FPVDs separately for each point p_i in the strip. By definition, each set S_i^α $(0 \leq \alpha \leq 2)$ contains the last but one points on the chains ending at p_i and having length $n_i - \alpha$. These sets and their FPVDs for every point up to p_i should be in hand before proceeding to process the point p_{i+1} in the strip.

The following lemmas are true for any horizontal strip of width 1.

Lemma 6.1.1. *[32] Let p_i, p_j, p_k, and p_l be four points lying inside a horizontal strip H of width 1 such that $x(p_i) < x(p_j) < x(p_k) < x(p_l)$. If p_i, p_j, p_k are pairwise independent and p_j, p_k, p_l are also pairwise independent, then p_i and p_l must be independent.*

Lemma 6.1.2. *Let $p_\ell \in S_i^\alpha$ (for some $0 \leq \alpha \leq 2$) be the farthest independent point of p_{i+1}. If p_ℓ, p_i, and p_{i+1} are mutually independent, then p_{i+1} is independent with all the points on the chain $C_{i,\ell}$ i.e., p_{i+1} is independent with every point in $S_{i,\ell}$.*

Proof. Let p_u be a point lying left to p_ℓ (i.e., $x(p_u) < x(p_\ell)$) on the chain $C_{i,\ell}$ (if p_u does not exist, the lemma is trivial). By the definition of $C_{i,\ell}$, the points p_u and p_i are independent and hence p_u, p_ℓ, p_i are pairwise independent. Therefore, by Lemma 6.1.1, p_u and p_{i+1} are independent as p_ℓ, p_i, and p_{i+1} are mutually independent. □

Lemma 6.1.3. *Let p_i and p_{i+1} be independent. Also, let p_u be the farthest independent point of p_{i+1} in S_i^α (for some $0 \leq \alpha \leq 2$). If p_u, p_i, and p_{i+1} are mutually independent then the cardinality of a GMIS (say M') of $\{p_1, p_2, \ldots, p_{i+1}\}$ having p_u, p_i, and p_{i+1} as right most three points is greater than or equal to the cardinality of a GMIS (say M'') of $\{p_1, p_2, \ldots, p_{i+1}\}$ having p_v, p_i, and p_{i+1} as right most three points, where p_v is the farthest point of p_{i+1} in S_i^β, for $0 \leq \alpha \leq \beta \leq 2$.*

Proof. $|M'| \geq |M''|$, follows from the definitions of S_i^α, S_i^β, and $\beta \geq \alpha$. Now we have to prove that p_v, p_i, and p_{i+1} are mutually independent. The case $\beta = \alpha$ is trivial. Let us consider the case $\beta > \alpha$ i.e., $\beta - \alpha = 1$ or 2. By the statement of the lemma, p_i and p_{i+1} are independent, and by the definition of S_i^β, p_v and p_i are independent. Now consider the chain $C_{i,u}$. Since p_u is the farthest point of p_{i+1} in S_i^α, the length of $C_{i,u}$ is $n_i - \alpha$. Let $\{s_1, s_2, \ldots, s_{n_i-\alpha-2}, s_{n_i-\alpha-1}(= p_u), s_{n_i-\alpha}(= p_i)\}$ be the set of points on the chain $C_{i,u}$ from left to right. Consider the following two cases:

Case A: $\beta - \alpha = 1$

Case B: $\beta - \alpha = 2$

In Case A, $s_{n_i-\alpha-2} \in S_i^\beta$ and in Case B, $s_{n_i-\alpha-3} \in S_i^\beta$.
Since $s_{n_i-\alpha-1}$ and p_{i+1} are independent, by Lemma 6.1.1, (i) $s_{n_i-\alpha-2}$ and p_{i+1} are independent, and also (ii) $s_{n_i-\alpha-3}$ and p_{i+1} are independent. This implies that p_v and p_{i+1} are independent as p_v is the farthest point of p_{i+1} in S_i^β. Thus the lemma follows. □

Lemma 6.1.4. *Let p_i and p_{i+1} be independent. If p_u is the farthest point of p_{i+1} in S_i^2, then p_u and p_{i+1} are independent.*

Proof. Note that $S_i^\alpha = \{p_j \in \mathcal{Q} \mid p_j \in S_{i,j}$ with $j < i$ and $|S_{i,j}| = n_i - \alpha\}$ for $\alpha = 0, 1, 2$. Consider a chain $C_{i,j}$ corresponds to $S_{i,j}$ such that $n(S_{i,j}) = n_i$. Let the members of the chain $C_{i,j}$ be $s_1, s_2, \ldots, s_{n_i-4}, s_{n_i-3}, s_{n_i-2}, s_{n_i-1}(= p_j), s_{n_i}(= p_i)$ from left to right. Now, consider the chain containing the points $s_1, s_2, \ldots, s_{n_i-4}, s_{n_i-3}, s_{n_i}(= p_i)$ and is of length $n_i - 2$. Therefore, $s_{n_i-3} \in S_i^2$. Observe that s_{n_i-3} is independent with p_{i+1}, since $s_{n_i-3}, s_{n_i-2}, s_{n_i-1}, s_{n_i}(= p_i)$ are pairwise independent and $x(s_{n_i-3}) < x(s_{n_i-2}) < x(s_{n_i-1}) < x(s_{n_i})$. Therefore, $x(s_{n_i}) - x(s_{n_i-3}) > 1$ (by Lemma 6.1.1). Note that, the points s_{n_i-3} and p_{i+1} are independent as $x(p_{i+1}) > x(p_i)$. Since s_{n_i-3} is independent with p_{i+1} implies that p_u is also independent with p_{i+1} as (i) $p_u \in S_i^2$, (ii) $s_{n_i-3} \in S_i^2$, and (iii) p_u is the farthest point of p_{i+1} in S_i^2. Thus the lemma follows. □

Lemma 6.1.5. *Let p_i and p_{i+1} be independent. If p_u, p_v, and p_w are the farthest points of p_{i+1} in S_i^0, S_i^1, and S_i^2 respectively, then either (i) p_u, p_{i+1}, or (ii) p_v, p_{i+1}, or (iii) p_w, p_{i+1} must be independent.*

Proof. Follows from Lemma 6.1.4 as p_w and p_{i+1} are independent. □

6.1.2.1 Algorithm to process a point

Let the current point to be processed is p_{i+1}. We assume that the set of points $\{p_1, p_2, \ldots, p_i\}$ are already processed one by one from left to right and we have in hand (i) the collection $\{S_{u,v}\}$ and $n(S_{u,v})$ for $1 \leq v < u \leq i$, (ii) the sets S_u^0, S_u^1, S_u^2 and their FPVDs for every $u \leq i$. Let $S_{i+1}^0 = S_{i+1}^1 = S_{i+1}^2 = \emptyset$. Now, we describe the method of processing the point p_{i+1} and the goal is to find a longest chain ending at p_{i+1}. If $d(p_i, p_{i+1}) > 1$, then we find a point $p_\ell \in S_i^0$, which is farthest from p_{i+1}. If $d(p_{i+1}, p_\ell) > 1$, then $S_{i+1,i} = \{p_{i+1}\} \cup S_{i,\ell}$ (i.e., we extend the chain ending at p_i corresponds to $S_{i,\ell}$ up to p_{i+1}) and $n(S_{i+1,i}) = n(S_{i,\ell}) + 1$. Note that, we are not storing $S_{i+1,i}$ explicitly. We can use a matrix M of size $m \times m$ and store p_ℓ in the $(i+1, i)^{th}$ entry of M. If $d(p_{i+1}, p_\ell) \leq 1$,

then we repeat the same process with S_i^1 and S_i^2 in order. So far, we tried to find a chain ending at p_{i+1} and having p_i and p_{i+1} as its two right most points, if exists. However, this chain need not be the longest chain ending at p_{i+1}. To find a longest chain ending at p_{i+1} we repeat the above process with $p_{i-1}, p_{i-2}, \ldots, p_1$ instead of p_i. Calculate $n_{i+1} = \max\{n(S_{i+1,j}) \mid j < i+1\}$. Observe that, n_{i+1} is length of the longest chain ending at p_{i+1}. Thus, we obtained a longest chain ending at p_{i+1}.

We use the sets S_{i+1}^{α} for $0 \leq \alpha \leq 2$, to store the last but one points on the chains of length $n_{i+1} - \alpha$ and ending at p_{i+1}, respectively. These sets are required to process the remaining points (i.e., the points right to p_{i+1}) in the strip. To find the sets S_{i+1}^0, S_{i+1}^1, and S_{i+1}^2, we repeat the entire process again as we are not storing all the chains ending at p_{i+1}, explicitly. If $n(S_{i+1,j}) = n_{i+1} - \alpha$ for any $j < i+1$, then $S_{i+1}^{\alpha} = S_{i+1}^{\alpha} \cup \{p_j\}$ for $0 \leq \alpha \leq 2$. Next, we store FPVDs of S_{i+1}^0, S_{i+1}^1, and S_{i+1}^2. The pseudo code of the algorithm for processing the point p_{i+1} is given in Algorithm 6.1. Boolean variables $flag1$ and $flag2$ are used in Algorithm 6.1 to handle the cases $S_i^{\alpha} = \emptyset$ for any $\alpha = 0, 1, 2$, and there is no point p_j in the strip such that $x(p_j) < x(p_{i+1})$ and $d(p_j, p_{i+1}) > 1$, respectively.

6.1.2.2 Correctness of Algorithm 6.1

Let the current point being processed is p_{i+1}. If $d(p_{i+1}, p_i) > 1$ we check for the independence of p_{i+1} with the farthest point in S_i^0, S_i^1, and S_i^2 in order. The farthest independent point, say p_ℓ, encountered first is considered to be part of the longest chain ending at p_{i+1}. The existence of p_ℓ is guaranteed by Lemma 6.1.5. By Lemma 6.1.2, p_{i+1} is independent with all the points on the chain $C_{i,\ell}$. Therefore, the set of points on the chain together with p_{i+1} form an independent set of $\{p_1, p_2, \ldots, p_{i+1}\}$ and is a possible maximum independent set having p_ℓ, p_i, and p_{i+1} as right most three points (see Lemma 6.1.3). Hence, we can safely extend the chain $C_{i,\ell}$ up to p_{i+1}. We tested all the points $p_i, p_{i-1}, \ldots, p_1$ (see line number 3 in Algorithm 6.1) for independence with p_{i+1} and hence we are considering all possible chains ending at p_{i+1}.

Algorithm 6.1 Processing_the_ point p_{i+1}

Require: (i) $S_{u,v}$ (in the form of matrix M) and $n(S_{u,v})$ for $1 \leq v < u \leq i$, and (ii) S_u^0, S_u^1, and S_u^2 and their FPVDs for $u \leq i$.

Ensure: (i) $S_{i+1,j}$ (in the form of matrix M) and $n(S_{i+1,j})$ for $j < i+1$, and (ii) $S_{i+1}^0, S_{i+1}^1, S_{i+1}^2$ and their FPVDs.

1: Let M be a matrix of size $m \times m$ and $M[i,j] \leftarrow \phi$ for $1 \leq i,j \leq m$
2: $flag1 = 0, flag2 = 0$
3: **for** $(w = i, i-1, \ldots, 1)$ **do**
4: **if** $(d(p_w, p_{i+1}) > 1)$ **then**
5: $flag2 = 1$
6: **for** $(\alpha = 0, 1, 2)$ **do**
7: **if** $(S_w^\alpha \neq \emptyset)$ **then**
8: $flag1 = 1$
9: Find the farthest point p_ℓ of p_{i+1} in $FPVD(S_w^\alpha)$ using planar point location algorithm [85].
10: **if** $(d(p_\ell, p_{i+1}) > 1)$ **then**
11: $M[i+1, w] \leftarrow p_\ell$
12: $n(S_{i+1,w}) = n(S_{w,\ell}) + 1$
13: **break** /* break the for loop **for** $(\alpha = 0, 1, 2)$ */
14: **end if**
15: **end if**
16: **end for**
17: **if** $(flag1 = 0)$ **then**
18: $M[i+1, w] \leftarrow p_w$
19: $n(S_{i+1,w}) = 2$
20: **end if**
21: **end if**
22: **end for**
23: $S_{i+1}^0 \leftarrow \emptyset, S_{i+1}^1 \leftarrow \emptyset, S_{i+1}^2 \leftarrow \emptyset$
24: **if** $flag2 = 0$ **then**
25: $n_{i+1} = 1$
26: **else**
27: $n_{i+1} = \max\{n(S_{i+1,j}) \mid j < i+1\}$
28: Repeat line numbers 3 - 22 by replacing lines 11 - 12 by lines 29 - 33.
29: **for** $(\alpha = 0, 1, 2)$ **do**
30: **if** $(n(S_{i+1,w}) = n_{i+1} - \alpha)$ **then**
31: $S_{i+1}^\alpha = S_{i+1}^\alpha \cup \{p_w\}$
32: **end if**
33: **end for**
34: Compute and store $FPVD(S_{i+1}^0), FPVD(S_{i+1}^1)$, and $FPVD(S_{i+1}^2)$.
35: **end if**

Lemma 6.1.6. *Algorithm 6.1 processes the point p_{i+1} correctly in $O(i \log i)$ time and uses $O(m^2)$ space, where m is the number of points lying in the strip.*

Proof. Correctness of Algorithm 6.1 follows from the discussion in Subsection 6.1.2.2. The worst case time complexity of lines 4 - 21 is $O(\log i)$ due to planar point location in line number 9. Therefore, time complexity of lines 3 - 22 is $O(i \log i)$. Again, time complexity of lines 27 - 33 is $O(i \log i)$. Computing FPVDs in line number 34 can be done in $O(i \log i)$. Thus the total time complexity of Algorithm 6.1 is $O(i \log i)$.

The space complexity follows from (i) size of the matrix M, (ii) collection $\{S_{i,j}\}$, (iii) counters $n(S_{i,j})$, (iv) sets S_i^0, S_i^1, S_i^2, and (v) storing FPVDs of the sets S_i^0, S_i^1, S_i^2 for $1 \leq i \leq m$. All together uses $O(m^2)$ space. □

6.1.2.3 Algorithm to find a GMIS for the points inside a strip

Algorithm to compute a GMIS for the set of points $\{p_1, p_2, \ldots, p_m\}$ within a strip H of width 1 is as follows : for a given predefined constant μ, we execute Algorithm 6.1 for each point $p_{\mu+1}, p_{\mu+2}, \ldots, p_m$ in the strip and report the largest set $S_{i,j}$ for $1 \leq j < i \leq m$. Hence, the size of a GMIS for a given strip of width 1 is equal to $\max\{n_1, n_2, \ldots, n_m\}$. The pseudo code of the algorithm is available in Algorithm 6.2.

Algorithm 6.2 Maximum_independent_set_ strip ($\mathcal{Q}$)

Require: The set $\mathcal{Q} = \{p_1, p_2, \ldots, p_m\}$ of m points lying in the strip H of width 1 and a constant μ.

Ensure: A maximum cardinality subset $\mathcal{Q}'$ of $\mathcal{Q}$ such that the points in $\mathcal{Q}'$ are mutually independent.

1: For the points $\{p_1, p_2, \ldots, p_\mu\}$ compute $\{S_{i,j}\}(j < i)$ in brute force manner.
2: **for** $(i = \mu + 1$ to $m)$ **do**
3: Process the point p_i by calling Algorithm 6.1.
4: **end for**
5: Return a set with maximum cardinality among $\{S_{i,j}\}$ for $1 \leq j < i \leq m$.

6.1.2.4 Correctness of Algorithm 6.2

Correctness of Algorithm 6.2 follows from, (i) for every point p_i in the strip Algorithm 6.2 computes a longest chain ending at p_i by calling Algorithm 6.1 and (ii) Algorithm 6.2 returns the longest chain after processing all the points in the strip.

Theorem 6.1.7. *Algorithm 6.2 correctly computes a GMIS for the set $\mathcal{Q} = \{p_1, p_2, \ldots, p_m\}$ of points inside a strip H of width 1 in $O(m^2 \log m)$ time using $O(m^2)$ space.*

Proof. Correctness of the algorithm follows from Lemma 6.1.6. Time complexity of Algorithm 6.2 is $\sum_{i=\mu+1}^{m} O(i \log i) + O(1)$ (see **for** loop in line number 2 in Algorithm 6.2, where it calls Algorithm 6.1 $O(m)$ times). Therefore, the total time complexity of Algorithm 6.2 is $O(m^2 \log m)$.

Space complexity of Algorithm 6.2 follows from Lemma 6.1.6 as we can reuse the matrix M for every call to Algorithm 6.1. □

6.1.2.5 Algorithm to find a GMIS for the given set $\mathcal{P}$

We partition the region containing the point set $\mathcal{P}$ into horizontal strips of width 1 such that no point of $\mathcal{P}$ lies on the boundary of any strip. Let $H_1, H_2, \ldots, H_\nu$ be the strips in the partition. We execute Algorithm 6.2 for every strip H_i for $1 \leq i \leq \nu$. Let $MIS_1, MIS_2, \ldots, MIS_\nu$ be the largest possible independent sets correspond to the points in $\mathcal{P} \cap H_1, \mathcal{P} \cap H_2, \ldots, \mathcal{P} \cap H_\nu$ respectively. Let MIS_{odd} and MIS_{even} be the union of maximum independent sets in odd and even strips, respectively. We report MIS_{odd} if $|MIS_{odd}| \geq |MIS_{even}|$, otherwise we report MIS_{even}. The pseudo code of the algorithm is given in Algorithm 6.3.

6.1.2.6 Correctness and analysis of Algorithm 6.3

Correctness of Algorithm 6.3 follows from the fact that MIS_{odd} and MIS_{even} are independent sets as all strips are of width 1 unit and the points in any two even strips (resp. odd) are independent.

Algorithm 6.3 Maximum_independent_set ($\mathcal{P}$)

Require: The set $\mathcal{P}$ of n points and strips $H_1, H_2, \ldots, H_\nu$.
Ensure: An independent subset of $\mathcal{P}$.

Compute $MIS_1, MIS_2, \ldots, MIS_\nu$ for the points lying in strips $H_1, H_2, \ldots, H_\nu$ by calling Algorithm 6.2 for each strip separately.
if $(\nu = 2u)$ **then**
    $MIS_{odd} = \bigcup_{i=0}^{u-1} MIS_{2i+1}$ and $MIS_{even} = \bigcup_{i=1}^{u} MIS_{2i}$
else
    if $(\nu = 2u + 1)$ **then**
        $MIS_{odd} = \bigcup_{i=0}^{u} MIS_{2i+1}$ and $MIS_{even} = \bigcup_{i=1}^{u} MIS_{2i}$
    end if
end if
if $|MIS_{odd}| \geq |MIS_{even}|$ **then**
    return MIS_{odd}
else
    return MIS_{even}
end if

Theorem 6.1.8. *Given a set $\mathcal{P}$ of n points correspond to a given UDG, a subset of at least $\frac{1}{2}|OPT|$ mutually independent points can be computed in $O(n^2 \log n)$ time and using $O(n^2)$ space using Algorithm 6.3, where $|OPT|$ is the cardinality of a largest independent set for the point set $\mathcal{P}$.*

Proof. Let χ be the solution obtained by Algorithm 6.3. We have to prove that $|\chi| > \frac{1}{2}|OPT|$, where OPT is a GMIS of $\mathcal{P}$. Since MIS_{odd} and MIS_{even} are union of solutions in odd and even strips respectively, we have,

$$|MIS_{odd}| + |MIS_{even}| \geq |OPT| \tag{6.1}$$

$$|\chi| + |\chi| \geq |MIS_{odd}| + |MIS_{even}| \tag{6.2}$$

From inequalities 6.1 and 6.2, $|\chi| \geq \frac{1}{2}|OPT|$.

The time and space complexities follow as we execute Algorithm 6.3 for every strip independently. Thus the theorem follows. □

6.2 Geometric Maximum Weighted Independent Set Problem

In this section, we present a 2-factor approximation algorithm for the GMWIS problem. The 2-factor approximation algorithm designed for the unweighted case is not straightforward to extended for the weighted case.

6.2.1 Data structures

Given a set $\mathcal{P}$ of n points in the plane, an *emptiness query* consists of determining whether a given *query range* R contains no point, that is, whether $\mathcal{P} \cap R = \emptyset$, returning an arbitrary point in $\mathcal{P} \cap R$ if exists. A *maximum query* is defined similarly, but the points have real weights and the query finds a point of maximum weight in $\mathcal{P} \cap R$, if it exists. In an *anti-disk* query, the range R is the complement of a disk of arbitrary radius r centered at q, that is, the region formed by all points with Euclidean distance greater than r from q.

Maximum queries can be reduced to emptiness queries as follows. As far as we know, this reduction has never been published, but it is likely that other researchers independently noticed the same construction.

Lemma 6.2.1. *Consider a set of n points with weights and a query range. A data structure for answering emptiness queries with storage $S(n) = \Omega(n)$, preprocessing time $B(n) = \Omega(n)$, and query time $Q(n)$ yields a data structure for answering maximum queries for the same range with preprocessing time $O(B(n) \log n)$, query time $O(Q(n) \log n)$, and storage $O(S(n) \log n)$.*

Proof. We assume that all weights are distinct, otherwise we break the ties arbitrarily. We build a binary tree where each node contains a data structure for emptiness queries as follows. The root contains the emptiness data structure for all points. Consider a node which stores the data structure for a subset of points of $\mathcal{P}$. Let μ_w be the median

weight among the points in the subset. The left subtree is defined recursively for the points in the subset with weight at most μ_w, and the right subtree is defined analogously for the remaining points of the subset. The recursion stops when the subset has only 1 point.

Notice that the tree has $\lceil \lg n \rceil + 1$ levels, and exactly n points at each level. The total storage is therefore

$$S'(n) = \sum_{\ell=0}^{\lceil \lg n \rceil + 1} 2^\ell S\left(\frac{n}{2^\ell}\right) \leq \sum_{\ell=0}^{\lceil \lg n \rceil + 1} S(n) = O(S(n) \log n),$$

where the first inequality uses that $S(n) = \Omega(n)$. The preprocessing time is calculated the same way.

To answer a maximum query, we verify at the root node whether the query range is empty. If not, we verify if the query range is empty for the right subtree. If it is empty, we recurse on the left subtree. Otherwise, we recurse on the right subtree. The procedure ends when we reach a leaf. The leaf reached is the point of maximum weight inside the query range, which we return. The bound on the query time follows from the fact that we performed $O(\log n)$ emptiness queries, one per level. □

Many techniques yield efficient data structures for anti-disk emptiness searching. For example, one can use a farthest point Voronoi diagram, point location, lifting (inversion), duality, etc. [11].

Lemma 6.2.2. *Given a set of n points, there exists a data structure that answers anti-disk emptiness queries in $O(\log n)$ time with $O(n)$ storage and $O(n \log n)$ preprocessing time.*

Combining Lemmas 6.2.1 and 6.2.2, we have:

Lemma 6.2.3. *Given a set of n weighted points, there exists a data structure that answers anti-disk maximum queries in $O(\log^2 n)$ time with $O(n \log n)$ and $O(n \log^2 n)$ storage and preprocessing time, respectively.*

Sometimes, the reduction from Lemma 6.2.1 is an overkill, and a logarithmic factor can be avoided by a simpler approach.

Lemma 6.2.4. *Given a set of n points with integer weights in $\{1, \ldots, n\}$, there exists a data structure that answers anti-disk maximum queries in $O((w_{max} - w_{ret}) \log n)$ time with $O(n)$ storage and $O(n \log n)$ preprocessing time, where w_{max} is the maximum weight among the points in the data structure and w_{ret} is the weight of the point returned by the query, or 0 if the query is empty.*

Proof. We keep an array of n data structures $D_1, \ldots, D_n$ from Lemma 6.2.2 where D_i stores the points of weight i. To answer a query, we start from the maximum i that corresponds to a non-empty D_i, decrementing i until the query returns non-empty. Analysis follows from the summation of individual complexities. □

Definition 6.2.5. A search problem is called *decomposable* if for any set S and $S' \subseteq S$, the answer to a query $q(S)$ can be calculated in constant time from the results of $q(S')$ and $q(S \setminus S')$.

Lemma 6.2.6 ([10, 79])**.** *A static data structure for a decomposable problem storing n elements with preprocessing time $B(n)$, query time $Q(n)$, and storage $S(n)$ yields a semi-dynamic data structure supporting insertions for the same problem with insertion time $O(\frac{B(n) \log n}{n})$, query time $O(Q(n) \log n)$, and storage $O(S(n) + \log n)$.*

Combining Lemma 6.2.6 with Lemmas 6.2.3 and 6.2.4 we have the following lemmas.

Lemma 6.2.7. *Given a set of n weighted points, there exists a semi-dynamic data structure that answers anti-disk* maximum *queries in $O(\log^3 n)$ time with $O(n \log n)$ storage and $O(\log^3 n)$ insertion time.*

Lemma 6.2.8. *Given a set of n points with integer weights in $\{1, \ldots, n\}$, there exists a semi-dynamic data structure that answers anti-disk* maximum *queries in $O((w_{max} - w_{ret}) \log^2 n)$ time with $O(n)$ storage and $O(\log^2 n)$ insertion time, where w_{max} is the maximum weight among the points in the data structure and w_{ret} is the weight of the point returned by the query, or 0 if the query is empty.*

6.3 The GMWIS Problem on Small Strips

In this section, we present a 2.16-factor approximation algorithm for the GMWIS problem. The following lemma shows why determining the maximum (weighted) independent set within a strip is useful for approximating the unrestricted problem. The proof is a simple application of the shifting strategy [55], paying attention to the fact that the width h must be a constant rational number for the proof to work

Lemma 6.3.1. *An exact algorithm for the maximum (weight) independent set for a set of m points inside a strip of rational width h with running time $T(m)$ yields an approximation algorithm for maximum (weight) independent set of n points with approximation ratio $1 + 1/h$ and running time $O(T(n) + n \log n)$.*

Matsui [71] showed that a unit disk graph inside a strip of width $\frac{\sqrt{3}}{2}$ is a co-comparability graph[1], and therefore the maximum weighted independent set can be determined exactly in time linear on the size of the input graph [64]. Since a complete graph is a unit disk graph within an arbitrarily small strip, building the graph takes quadratic time in the worst case. The crucial property shown by Matsui and used by our algorithm is the following.

Lemma 6.3.2. *Let H be a strip of width $\frac{\sqrt{3}}{2}$. Consider $p_i, p_j, p_k \in H$ with $x(p_k) < x(p_j) < x(p_i)$. If $d(p_i, p_j) > 1$ and $d(p_j, p_k) > 1$, then $d(p_i, p_k) > 1$.*

To obtain a logarithmic improvement for the unweighted version, we use an additional property of unit disk graphs from [70].

Definition 6.3.3. The *left neighborhood* of a point p is the set of points q with $d(p, q) \leq 1$ and $x(q) \leq x(p)$.

Lemma 6.3.4. *[70] The left neighborhood of a point p has at most 3 independent vertices.*

[1]An undirected graph is said to be a comparable graph, if each edge in the graph can be oriented such that the vertices in the oriented graph satisfy anti-symmetry and transitivity properties with respect to adjacency. A graph is said to be a co-comparable graph, if its complement is a comparable graph.

Let $\mathcal{Q} = \{p_1, p_2, \ldots, p_m\}$ be the set of points lying in a strip H with $p_1, p_2, \ldots, p_m$ in increasing order of their x-coordinates. Given $1 \leq i \leq m$, we define the set $S_i \subseteq \mathcal{Q}$ is as follows: (i) S_i is an independent set, (ii) p_i is the rightmost point of S_i, and (iii) S_i is a maximum weight set satisfying the previous two properties. Let W_i denote the sum of the weights of the points in S_i. For simplicity, the sets S_i can be viewed as x-monotone polygonal chains formed by connecting the points of S_i from left to right (see Figure 6.3). The objective of the proposed algorithm is to extend these chains as long as possible. A chain containing maximum number of points is reported.

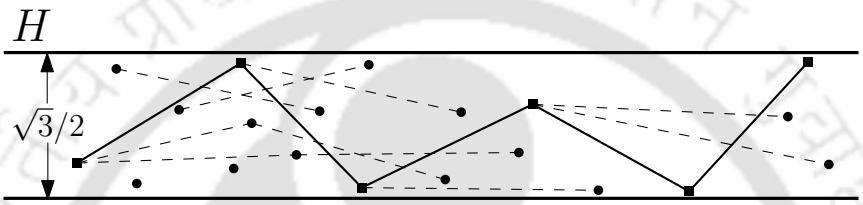


Figure 6.3: Pictorial representation of the sets S_i in the form of chains, with the maximum independent set marked.

The proposed algorithm uses dynamic programming to compute S_i for $i = 1, \ldots, m$ as follows. We calculate S_i using S_j for $j < i$ iteratively as $S_i = S_j \cup \{p_i\}$, where j is the index $j < i$ that satisfies $d(p_j, p_i) > 1$ and maximizes W_j. In case there is no point p_j, satisfying $d(p_j, p_i) > 1$ and $j < i$, the algorithm sets $S_i = \{p_i\}$. To efficiently determine the index j, we use the data structure from Lemma 6.2.7 (weighted) or 6.2.8 (unweighted). The pseudo-code is presented in Algorithm 6.4.

Theorem 6.3.5. *Algorithm 6.4 correctly computes an MWIS for the set $\mathcal{Q} = \{p_1, p_2, \ldots, p_m\}$ inside a strip of width $\frac{\sqrt{3}}{2}$ in $O(m \log^3 m)$ time using $O(m \log m)$ space. In the unweighted version, the complexities decrease to $O(m \log^2 m)$ time and $O(m)$ space.*

Proof. To see that S_i is calculated correctly by the procedure, notice that by Lemma 6.3.2, $d(p_i, p_j) > 1$ for $j < i$ implies that a point p_k to the left of p_j with $d(p_j, p_k) > 1$ satisfies $d(p_i, p_k) > 1$. Note that by construction, S_i contains p_i and its weight is maximum.

Algorithm 6.4 Small_strip($\mathcal{Q}$)

Require: The set $\mathcal{Q}$ of m points $p_1, \ldots, p_m$ ordered by x coordinate inside a strip of width $\frac{\sqrt{3}}{2}$.

Ensure: A maximum weight independent set of $\mathcal{Q}$.

1: D = data structure from Lemma 6.2.7 (weighted) or 6.2.8 (unweighted)
2: S = array $[1 \ldots m]$ of integers
3: W = array $[0 \ldots m]$ of reals (weighted) or integers (unweighted)
4: $W[0] = 0$, $max = 1$
5: **for** $i = 1, \ldots, m$ **do**
6: A = complement of the disk of radius 1 centered at p_i
7: $j =$ `maxQuery`(D, A) ▷ Returns 0 if no point is found
8: $S[i] = j$ ▷ Represents $S_i = S_j \cup \{p_i\}$ with $S_0 = \emptyset$
9: $W[i] = W[j] + w(p_i)$
10: `insert`$(D, i, p_i, W[i])$ ▷ Insert point p_i with label i and weight $W[i]$ into D
11: **if** $W[i] > W[max]$ **then**
12: $max = i$
13: **end if**
14: **end for**
15: $i = max$, $IS = \emptyset$
16: **while** $i \neq 0$ **do**
17: $IS = IS \cup \{p_i\}$
18: $i = S[i]$
19: **end while**
20: **return** IS

To execute the algorithm in $O(m \log^3 m)$ time, we use the data structure for anti-disk maximum queries supporting insertions from Lemma 6.2.7 to store p_j for $j < i$. The weight of p_j in the data structure is W_j. The $O(m)$ insertions (line 10) and queries (line 7) take together $O(m \log^3 m)$ time, and the space is $O(m \log m)$ due to the storage of the data structure. The algorithm stores S_i as a linked list from right to left. Part of the linked lists are shared by different sets S_i, forming a directed acyclic graph.

To reduce the running time to $O(m \log^2 m)$ for the unweighted version, we use the data structure from Lemma 6.2.8. In the unweighted version, $W_i = |S_i|$ and therefore the weight W_i is an integer between 1 and m. Let max be the index of the point of maximum weight in the data structure at the time when a point of index j is returned by the `maxQuery` sub-routine (see line number 7 in Algorithm 6.4). We use Lemma 6.3.4

to show that $W_{max} - W_j \leq 3$. If $|S_{max}| \leq 3$, the statement follows trivially. Otherwise, let p_a, p_b, p_c, p_{max} be the four rightmost elements in S_{max} from left to right. We have $W_{max} = 1 + W_c = 2 + W_b = 3 + W_a$. By Lemma 6.3.4, p_a, p_b, p_c, p_{max} cannot be all adjacent to p_i, which shows that $w_{max} - w_{ret} \leq 3$. Therefore, using the data structure from Lemma 6.2.8, the $O(m)$ insertions and queries take together $O(m \log^2 m)$ time, and the space is $O(m)$. □

Combining Lemma 6.3.1 and Theorem 6.3.5, we obtain the following theorem.

Theorem 6.3.6. *Given a set $\mathcal{P}$ of n weighted points, we can compute an approximation to the GMWIS problem in $O(n \log^3 n)$ time and $O(n \log n)$ space with approximation ratio $(1 + \frac{2}{\sqrt{3}}) < 2.16$. In the unweighted version, the complexities decrease to $O(n \log^2 n)$ time and $O(n)$ space.*

6.4 The GMWIS Problem on Large Strips

In this section, we present a 2-factor approximation algorithm for the GMWIS problem. The algorithm uses a decomposition of the region containing the points into strips of width 1.

Let $\mathcal{Q} = \{p_1, p_2, \ldots, p_m\}$ be the set of points lying in the strip H with $p_1, p_2, \ldots, p_m$ in increasing order of their x-coordinates. Given $1 \leq j < i \leq m$ with $d(p_i, p_j) > 1$, we define the set $S_{i,j} \subseteq \mathcal{Q}$ as follows: (i) $S_{i,j}$ is an independent set, (ii) p_j and p_i are the two rightmost points of $S_{i,j}$, and (iii) $S_{i,j}$ is a maximum weight set satisfying the previous two properties. Furthermore, we define $S_{i,0} = \{p_i\}$. Let $W_{i,j}$ denote the sum of the weights of the points in $S_{i,j}$. Let $\mathcal{S}_i = \{S_{i,j} \mid 1 \leq j < i \text{ and } d(p_i, p_j) > 1\}$ denote the collections of sets $S_{i,j}$ for fixed i.

The proposed algorithm uses dynamic programming to compute $S_{i,j}$ for $i = 2, \ldots, m$ and $j = 1, 2, \ldots, i-1$ with $d(p_i, p_j) > 1$ as follows. We calculate $S_{i,j}$ using $S_{j,k}$ for $k < j < i$ iteratively as $S_{i,j} = S_{j,k} \cup \{p_i\}$, where k is the index $\ell < j$ maximizing $W_{j,\ell}$ and satisfying the constraint that $\{p_i, p_j, p_\ell\}$ is an independent set. In case there is no point p_ℓ satisfying the previous constraint, the algorithm sets $S_{i,j} = \{p_i, p_j\}$. To

efficiently determine the index $\ell < j$ that satisfies the previous constraint and maximizes W_ℓ we use an array of data structures $D_1, \ldots, D_m$ from Lemma 6.2.3 (weighted) or 6.2.4 (unweighted). The data structure D_i stores a point p_j with weight $W_{i,j}$ for each set $S_{i,j} \in \mathcal{S}_i$. Note that the data structure D_j only stores points p_k such that $d(p_j, p_k) > 1$. The pseudo-code is presented in Algorithm 6.5.

Theorem 6.4.1. *Algorithm 6.5 correctly computes an MWIS for the set of points $\mathcal{Q} = \{p_1, p_2, \ldots, p_m\}$ inside a strip of width 1 in $O(m^2 \log^2 m)$ time using $O(m^2 \log m)$ space. In the unweighted version, the complexities decrease to $O(m^2 \log m)$ time and $O(m^2)$ space.*

Proof. To see that $S_{i,j}$ calculated by the procedure is indeed an independent set, notice that by Lemma 6.1.1, $\{p_i, p_j, p_k\}$ be independent implies that a point p_s to the left of p_k with $\{p_j, p_k, p_s\}$ independent satisfies $d(p_i, p_s) > 1$. Note that by construction, $S_{i,j}$ contains p_i, p_j as the points with maximum x coordinates and its weight is maximum.

To execute the algorithm in $O(m^2 \log^2 m)$ time, we use m data structures for anti-disk maximum queries from Lemma 6.2.3. For each point p_i, the data structure D_i stores the point p_j for each set $S_{i,j} \in \mathcal{S}_i$. The weight associated with p_j is $W_{i,j}$. The data structure D_i is static, preprocessed (line 25) with points collected in the set T. The time to preprocess all $O(m)$ data structures from the corresponding sets T is therefore $O(m) \cdot O(m \log^2 m)$. The point p_k of maximum weight satisfying $k < j$ and $\{p_i, p_j, p_k\}$ independent is determined with a single anti-disk maximum query (line 16) on the data structure D_j associated with $\mathcal{S}_j$. All the $O(m^2)$ queries take together $O(m^2) \cdot O(\log^2 m) = O(m^2 \log^2 m)$ time. Space is dominated by the $O(m)$ data structures with $O(m \log m)$ storage each.

To reduce the running time to $O(m^2 \log m)$ and the space to $O(m^2)$ for the unweighted version, we use the data structure from Lemma 6.2.4, and the same argument as in the proof of Theorem 6.3.5 shows that $w_{max} - w_{ret} \leq 3$. □

Combining Lemma 6.3.1 and Theorem 6.4.1, we obtain the following theorem.

Theorem 6.4.2. *Given a set $\mathcal{P}$ of n weighted points, we can compute a 2-approximation to the GMWIS problem in $O(n^2 \log^2 n)$ time and $O(n^2 \log n)$ space. In the unweighted version, the complexities decrease to $O(n^2 \log n)$ time and $O(n^2)$ space.*

Algorithm 6.5 Large_strip($\mathcal{Q}$)

Require: The set $\mathcal{Q}$ of m points $p_1, \ldots, p_m$ in sorted order inside a strip of width 1.
Ensure: A maximum weight independent set of $\mathcal{Q}$.

D = array $[1 \ldots m]$ of data structure from Lemma 6.2.3 (weighted) or 6.2.4 (unweighted)
S = array $[1 \ldots m, 0 \ldots m]$ of integers
W = array $[1 \ldots m, 0 \ldots m]$ of reals (weighted) or integers (unweighted)
$max = (1, 0)$
for $i = 1, \ldots, m$ **do**
    $W[i, 0] = w(p_i)$, $S[i, 0] = 0$ ▷ $S[i, 0] = 0$ represents $S_{i,0} = \{p_i\}$
    if $W[i, 0] > W[max]$ **then**
        $max = (i, 0)$
    end if
end for
for $i = 2, \ldots, m$ **do**
    $T = \emptyset$
    for $j = 1, \ldots, i - 1$ **do**
        if $d(p_i, p_j) > 1$ **then**
            A = complement of the disk of radius 1 centered at p_i
            $k =$ `maxQuery`$(D[j], A)$ ▷ Returns 0 if no point is found
            $S[i, j] = k$ ▷ Represents $S_{i,j} = S_{j,k} \cup \{p_i\}$
            $W[i, j] = W[j, k] + w(p_i)$
            $T = T \cup \{(j, p_j, W[i, j])\}$ ▷ (label, point, weight) for $D[i]$
            if $W[i, j] > W[max]$ **then**
                $max = (i, j)$
            end if
        end if
    end for
    $D[i] =$ `preprocess`(T) ▷ New data structure from set T
end for
$(i, j) = max$, $IS = \{p_i\}$
while $j \neq 0$ **do**
    $IS = IS \cup \{p_j\}$
    $i = j$, $j = S[i, j]$
end while
return IS

6.5 A PTAS for the GMIS Problem

Let $\mathcal{R}$ be an enclosing rectangle of the point set $\mathcal{P}$ in the plane. Our PTAS is based on the shifting strategy, proposed by Hauchbaum and Maass [55]. In our PTAS we use two level shifting strategy. In the first level of shifting strategy, we execute $k+1$ iterations as follows: in the i-th iteration ($0 \leq i \leq k$), we partition the region $\mathcal{R}$ into disjoint vertical slabs such that (i) the first slab is of width i starting from left, (ii) width of each even slab is 1, and (iii) width of other slab is k (note that the width of the last slab may be less than k).

In an iteration of the first level, we consider only those vertical slabs containing at least one point in $\mathcal{P}$, and compute a maximum independent set by applying second level shifting strategy by considering horizontal partition of each vertical slab, add up the solutions of all slabs to get the solution of that iteration. The iteration producing maximum size solution is reported.

Lemma 6.5.1. *[32] If n_k is the maximum number of mutually independent points in a strip of width $k > 1$ and are within a $\frac{1}{2}$ distance from a vertical line ℓ, then $n_k \leq \frac{7k}{3} + 2$.*

6.5.1 Computing MIS for the points in a $k \times k$ square

Let $Q \subseteq \mathcal{P}$ be the set of points inside a cell χ of size $k \times k$. Consider a vertical line ℓ_v and a horizontal line ℓ_h that partition χ into four sub-cells each of size $\frac{k}{2} \times \frac{k}{2}$. Let $Q(\ell_v, \ell_h) \subseteq Q$ be the set of points whose distance from ℓ_v or ℓ_h is at most $\frac{1}{2}$, and $Q_1, Q_2, Q_3, Q_4 \subseteq Q$ be the set of points in the four quadrants whose distance from ℓ_v and ℓ_h is greater than $\frac{1}{2}$. To compute an MIS for the set of points in Q, we use the following divide and conquer technique.

Consider all possible subsets $Q' \subseteq Q(\ell_v, \ell_h)$ of size at most $2 \times n_k$, where $n_k = \frac{7k}{3} + 2$ (since $2 \times n_k$ is the maximum possible size of the point set in $Q(\ell_v, \ell_h)$ that can appear in an optimal solution due to Lemma 6.5.1). For each of Q', we do the following in each quadrant: delete all the points in Q_i ($i = 1, 2, 3, 4$) which are not independent

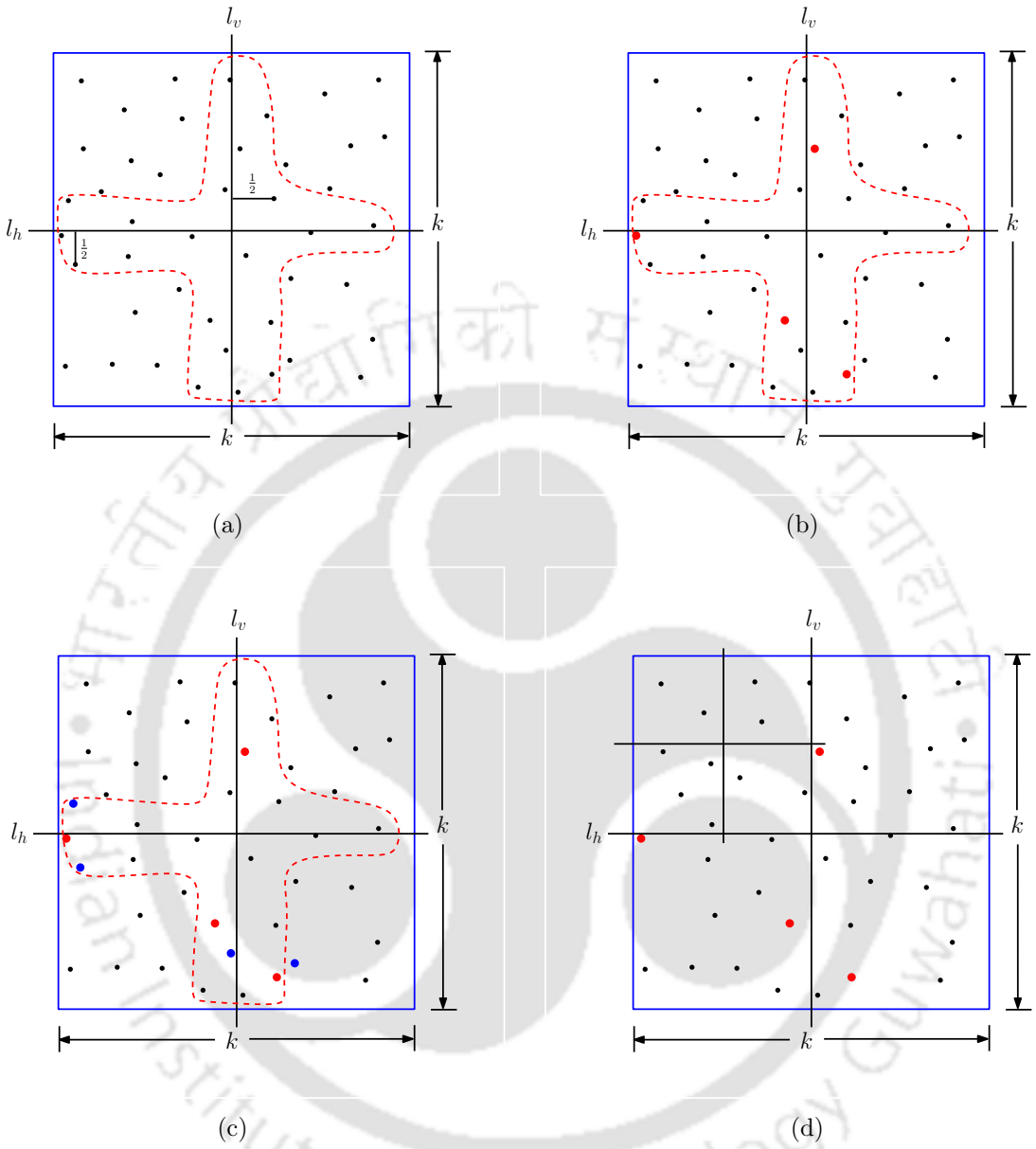


Figure 6.4: (a) The set of points within a $\frac{1}{2}$ distance from the lines ℓ_h and ℓ_v (shown in a loop), (b) a subset (Q') of mutually independent points (shown in red), (c) the subset of points that are not independent with the points in Q' (shown in blue), and (d) applying the recursive procedure after deleting the non-independent points.

with Q'. Let $Q'_i \subseteq Q_i$ be the remaining set of points. Now, compute the optimum solution for Q'_i recursively using the same procedure. This process is illustrated with

an example in Figure 6.4. If $T(m,k)$ is the time complexity for finding MIS in χ, then $T(m,k) = 4 * T(m, \frac{k}{2}) \times m^{2n_k} = m^{O(k)}$. Thus, we have the following result:

Theorem 6.5.2. *Given a set P of n points in the plane and an integer $k > 1$, the proposed algorithm computes an independent set of size at least $\frac{1}{(1+\frac{1}{k})^2}|OPT|$ in $n^{O(k)}$ time, where $|OPT|$ is the optimum size of the solution.*

6.6 Conclusion

In this chapter, we proposed a 2-factor approximation algorithm for the GMIS problem. The algorithm runs in $O(n^2 \log n)$ time and uses $O(n^2)$ space, outperforming the existing algorithms in the literature with respect to time complexity by a factor of $\frac{n}{\log n}$ [32]. We also proposed a PTAS for the same problem. The running time of our proposed PTAS is $n^{O(k)}$. The previous best known PTAS runs in $n^{O(k \log k)}$ time [32]. The proposed PTAS has advantage over [32] and [21] in terms of time and space complexities, respectively. The PTAS proposed in [32] runs in $n^{O(k \log k)}$ time and uses $O(n+k \log k)$ space. Similarly, the PTAS proposed in [21] has $n^{O(\frac{1}{\epsilon})}$ time and space. Our PTAS runs in $n^{O(k)}$ time and uses $O(n + k \log k)$ space.

For the GMWIS problem we proposed 2.16 and 2-factor approximation algorithms. The proposed 2.16-factor approximation algorithm runs in $O(n \log^3 n)$ time and uses $O(n \log n)$ space. The best known algorithm runs in $O(n^2)$ time [71]. In the unweighted version, the complexities decrease to $O(n \log^2 n)$ time and uses $O(n)$ space. The proposed 2-factor approximation algorithm runs in $O(n^2 \log^2 n)$ time and $O(n^2 \log n)$ space. In the unweighted version, the complexities decrease to $O(n^2 \log n)$ time and $O(n^2)$ space.

Chapter 7

Conclusions and Future Work

Most of the variants of dominating set problem are NP-hard in general graphs and they do not even admit constant factor approximation algorithms. This motivated researchers to study dominating set and its variants on other graph classes. In this thesis, we have studied minimum dominating set problem and some of its variants on unit disk graphs, namely, geometric minimum dominating set (GMDS) problem, minimum connected dominating set (MCDS) problem, geometric minimum liar's dominating set (GMLDS) problem and geometric maximum (weighted) independent set (GMIS/GMWIS) problem. Unfortunately, these problems are NP-hard on unit disk graphs too, but unlike in general graphs, they admit constant factor approximation algorithms and approximation schemes, thus quest for faster approximation algorithms encouraged us to study these problems.

This chapter aims to summarize the contribution made in this thesis and state some of the problems which might be carried as future research.

For the GMDS problem, we proposed a series of constant factor approximation algorithms. We first presented a simple $O(n \log k)$ time 5-factor approximation algorithm, where k is the size of the output. Next, we presented 4-factor and 3-factor approximation algorithms in $O(n^6 \log n)$ and $O(n^{11} \log n)$ time respectively, which improved the time complexities of best known result by a factor of $O(n^2)$ and $O(n^4)$ respectively [34]. We

have also presented $\frac{14}{3}$-factor and $\frac{45}{13}$-factor approximation algorithms in time $O(n^5 \log n)$ and $O(n^{10} \log n)$, respectively. Finally, we proposed a two-level shifting lemma and using this lemma we presented a $\frac{5}{2}$-factor approximation algorithm and a PTAS for the GMDS problem. Though the approximation algorithms presented in this thesis are best till today, yet there is a scope to improve the running times.

For the MCDS problem, we designed a distributed approximation algorithm with approximation factor 104 and has time and message complexities $O(\Delta)$ and $O(n)$, respectively. We also proposed a scheduling scheme that obtains $O(\Delta)$ conflict-free time slots to deal with interference. The proposed algorithm is the first constant factor approximation algorithm to achieve time complexity $O(\Delta)$. If the degree, Δ, of the given UDG is bounded by a constant, our algorithm runs in $O(1)$ time and produces a solution whose size is at most a constant multiple of the size of an optimal solution, regardless of the number of nodes in the graph. Whereas the existing constant factor approximation algorithms run in $O(n)$ time even if Δ is bounded by a constant. One can think of improving the approximation ratio while maintaining the time complexity $O(\Delta)$. As far as we know no PTAS is available for the weighted version of the MCDS problem on unit disk graphs. As future work, we consider designing a PTAS for the minimum weighted connected dominating set problem.

We introduced the liar's dominating set problem in the geometric setting. We showed that the GMLDS problem is NP-hard. Therefore, liar's dominating set problem is also NP-hard for UDGs, where geometric information (i.e. positions of the disks) are unknown. We designed constant factor approximation algorithms by extending the ideas used for the GMDS problem. Finally, we presented a PTAS. The GMLDS problem is considered for the case only one intruder and at most one device can lie (misreport) in the closed neighborhood of an intruder location. However, in real life scenario intrusion of more than one intruder and multiple devices can misreport. No results are available in the literature for the general case in general graphs. As a future direction one can think of studying the general km-MLDS problem, where k is the number of intruders

and m is the number of devices allowed to misreport.

We proposed a 2-factor approximation algorithm for the GMIS problem. The proposed algorithm runs in $O(n^2 \log n)$ time and uses $O(n^2)$ space, outperforming the existing algorithms in the literature with respect to time complexity by a factor of $\frac{n}{\log n}$ [32]. We also proposed a PTAS for the same problem. The running time of our proposed PTAS is $n^{O(k)}$, where k is an integer. The previous best known PTAS runs in $n^{O(k \log k)}$ time [32]. For the GMWIS problem we proposed 2.16 and 2-factor approximation algorithms. The proposed 2.16-factor approximation algorithm runs in $O(n \log^3 n)$ time and uses $O(n \log n)$ space. The best known algorithm runs in $O(n^2)$ time [71]. In the unweighted version, the complexities decrease to $O(n \log^2 n)$ time and uses $O(n)$ space. The proposed 2-factor approximation algorithm runs in $O(n^2 \log^2 n)$ time and $O(n^2 \log n)$ space. In the unweighted version, the complexities decrease to $O(n^2 \log n)$ time and $O(n^2)$ space. We would like to design an algorithm that improves both the time and space complexities.

We would like to introduce and/or improve the existing results for many other dominating set problems in other intersection graphs of disks of arbitrary radius, rectangles, convex polygons, fat objects, etc., as part of the future research.

Bibliography

[1] Pankaj K. Agarwal, Mark van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Computational Geometry*, 11(3):209–218, 1998.

[2] Abdollah Alimadadi, Mustapha Chellali, and Doost Ali Mojdeh. Liar's dominating sets in graphs. *Discrete Applied Mathematics*, 211:204–210, 2016.

[3] Robert B. Allan and Renu Laskar. On domination and independent domination numbers of a graph. *Discrete mathematics*, 23(2):73–76, 1978.

[4] Khaled M. Alzoubi, Peng-Jun Wan, and Ophir Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *Proceedings of the 3rd ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2002, June 9-11, 2002, Lausanne, Switzerland*, pages 157–164, 2002.

[5] Christoph Ambühl, Thomas Erlebach, Matús Mihalák, and Marc Nunkesser. Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs. In *Proceedings of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2006) and 10th International Workshop on Randomization and Computation (RANDOM 2006)*, LNCS 4110, pages 3–14, 2006.

[6] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy.

Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998.

[7] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM (JACM)*, 45(1):70–122, 1998.

[8] Gill Barequet, Matthew Dickerson, and Petru Pau. Translating a convex polygon to contain a maximum number of points. *Computational Geometry*, 8(4):167–179, 1997.

[9] Manjanna Basappa, Rashmisnata Acharyya, and Gautam K. Das. Unit disk cover problem in 2d. *Journal of Discrete Algorithms*, 33:193–201, 2015.

[10] Jon Louis Bentley and James B. Saxe. Decomposable searching problems I: Static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.

[11] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: Algorithms and applications, 3rd Edition*. Springer, 2008.

[12] Therese Biedl and Goos Kant. A better heuristic for orthogonal graph drawings. *Computational Geometry*, 9(3):159–180, 1998.

[13] Jeremy Blum, Min Ding, Andrew Thaeler, and Xiuzhen Cheng. Connected dominating set in sensor networks and MANETs. In *Handbook of combinatorial optimization*, pages 329–369. Springer, 2005.

[14] Heinz Breu and David G. Kirkpatrick. Unit disk graph recognition is NP-hard. *Computational Geometry*, 9(1):3–24, 1998.

[15] Hervé Brönnimann, John Iacono, Jyrki Katajainen, Pat Morin, Jason Morrison, and Godfried Toussaint. Space-efficient planar convex hull algorithms. *Theoretical Computer Science*, 321(1):25–40, 2004.

[16] Sergiy Butenko, Sera Kahruman-Anderoglu, and Oleksii Ursulenko. On connected domination in unit ball graphs. *Optimization Letters*, 5(2):195–205, 2011.

[17] Gruia Călinescu, Ion I. Mandoiu, Peng-Jun Wan, and Alexander Z. Zelikovsky. Selecting forwarding neighbors in wireless ad hoc networks. *Mobile Networks and Applications*, 9(2):101–111, 2004.

[18] Paz Carmi, Matthew J. Katz, and Nissan Lev-Tov. Covering points by unit disks of fixed location. In *Proceedings of the 18th International Symposium on Algorithms and Computation (ISAAC 2007)*, LNCS 4835, pages 644–655. Springer, 2007.

[19] Paz Carmi, Matthew J. Katz, and Nissan Lev-Tov. Polynomial-time approximation schemes for piercing and covering with applications in wireless networks. *Computational Geometry*, 39(3):209–218, 2008.

[20] Márcia R. Cerioli, Luerbio Faria, Talita O. Ferreira, and Fábio Protti. On minimum clique partition and maximum independent set on unit disk graphs and penny graphs: complexity and approximation. *Electronic Notes in Discrete Mathematics*, 18:73–79, 2004.

[21] Timothy M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *Journal of Algorithms*, 46(2):178–189, 2003.

[22] Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012.

[23] Xiuzhen Cheng, Min Ding, David Hongwei Du, and Xiaohua Jia. Virtual backbone construction in multihop ad hoc wireless networks. *Wireless Communications and Mobile Computing*, 6(2):183–190, 2006.

[24] Xiuzhen Cheng, Xiao Huang, Deying Li, Weili Wu, and Ding-Zhu Du. A

polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. *Networks*, 42(4):202–208, 2003.

[25] Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on Computing*, 14(1):210–223, 1985.

[26] Miroslav Chlebík and Janka Chlebíková. The complexity of combinatorial optimization problems on d-dimensional boxes. *SIAM Journal on Discrete Mathematics*, 21(1):158–169, 2007.

[27] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete mathematics*, 86(1-3):165–177, 1990.

[28] Francisco Claude, Gautam K. Das, Reza Dorrigiv, Stephane Durocher, Robert Fraser, Alejandro López-Ortiz, Bradford G. Nickerson, and Alejandro Salinger. An improved line-separable algorithm for discrete unit disk cover. *Discrete Mathematics, Algorithms and Applications*, 2(1):77–87, 2010.

[29] Decheng Dai and Changyuan Yu. A $5 + \epsilon$-approximation algorithm for minimum weighted dominating set in unit disk graph. *Theoretical Computer Science*, 410(8):756–765, 2009.

[30] Fei Dai and Jie Wu. On constructing k-connected k-dominating set in wireless ad hoc and sensor networks. *Journal of parallel and distributed computing*, 66(7):947–958, 2006.

[31] Bevan Das and Vaduvur Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *Proceeding of the 1997 IEEE International Conference on Communications (ICC 1997)*, volume 1, pages 376–380. IEEE, 1997.

[32] Gautam K. Das, Minati De, Sudeshna Kolay, Subhas C. Nandy, and Susmita Sur-Kolay. Approximation algorithms for maximum independent set of a unit disk graph. *Information Processing Letters*, 115(3):439–446, 2015.

[33] Gautam K Das, Robert Fraser, Alejandro Lóopez-Ortiz, and Bradford G Nickerson. On the discrete unit disk cover problem. *International Journal of Computational Geometry & Applications*, 22(05):407–419, 2012.

[34] Minati De, Gautam K. Das, Paz Carmi, and Subhas C. Nandy. Approximation algorithms for a variant of discrete piercing set problem for unit disks. *International Journal of Computational Geometry & Applications*, 23(06):461–477, 2013.

[35] Ding-Zhu Du, My T. Thai, Yingshu Li, Dan Liu, and Shiwei Zhu. Strongly connected dominating sets in wireless sensor networks with unidirectional links. In *Proceedings of the 8th Asia-Pacific Web Conference on Frontiers of WWW Research and Development - APWeb 2006*, LNCS 3841, pages 13–24, 2006.

[36] Ding-Zhu Du and Peng-Jun Wan. *Connected dominating set: theory and applications*, volume 77. Springer Science & Business Media, 2012.

[37] Yingfan L. Du and Hongmin W. Du. A new bound on maximum independent set and minimum connected dominating set in unit disk graphs. *Journal of Combinatorial Optimization*, 30(4):1173–1179, 2015.

[38] Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM Journal on Computing*, 34(6):1302–1323, 2005.

[39] Thomas Erlebach and Erik Jan Van Leeuwen. Domination in geometric intersection graphs. In *Proceedings of the 8th Latin American Symposium on Theoretical Informatics (LATIN 2008)*, LNCS 4957, pages 747–758. Springer, 2008.

[40] Uriel Feige. Approximating maximum clique by removing subgraphs. *SIAM Journal on Discrete Mathematics*, 18(2):219–225, 2004.

[41] Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy.

Interactive proofs and the hardness of approximating cliques. *Journal of the ACM (JACM)*, 43(2):268–292, 1996.

[42] Guilherme D. da Fonseca, Celina M. H. de Figueiredo, Vinícius G. Pereira de Sá, and Raphael C. S. Machado. Efficient sub-5 approximations for minimum dominating sets in unit disk graphs. *Theoretical Computer Science*, 540:70–81, 2014.

[43] Robert Fraser and Alejandro López-Ortiz. The within-strip discrete unit disk cover problem. In *Canadian Conference on Computational Geometry*, pages 53–58, 2012.

[44] Stefan Funke, Alexander Kesselman, Ulrich Meyer, and Michael Segal. A simple improved distributed algorithm for minimum CDS in unit disk graphs. *ACM Transactions on Sensor Networks (TOSN)*, 2(3):444–453, 2006.

[45] Bo Gao, Yuhang Yang, and Huiye Ma. A new distributed approximation algorithm for constructing minimum connected dominating set in wireless ad hoc networks. *International Journal of Communication Systems*, 18(8):743–762, 2005.

[46] Shenyong Gao and Ying Zhang. An improved distributed approximation algorithm for minimum connected dominating set. In *Proceeding of the 8th International Conference on Natural Computation (ICNC 2012)*, pages 1019–1022. IEEE, 2012.

[47] Michael R. Garey and David S. Johnson. The complexity of near-optimal graph coloring. *Journal of the ACM (JACM)*, 23(1):43–49, 1976.

[48] Michael R. Garey and David S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. Freeman, 1979.

[49] Matt Gibson and Imran A. Pirwani. Algorithms for dominating set in disk graphs: Breaking the $\log n$ barrier. In *Proceedings of the 18th Annual European Symposium on European Symposium on Algorithms (ESA 2010)*, LNCS 6346, pages 243–254. Springer, 2010.

[50] Sudipto Guha and Samir Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.

[51] Magnús M. Halldórsson. Approximating discrete collections via local improvements. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 160–169. SIAM, 1995.

[52] Sariel Har-Peled. *Geometric approximation algorithms*, volume 173. American mathematical society Boston, 2011.

[53] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS 1996)*, pages 627–636. IEEE, 1996.

[54] Teresa W. Haynes, Stephen Hedetniemi, and Peter Slater. *Fundamentals of domination in graphs*. CRC Press, 1998.

[55] Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM (JACM)*, 32(1):130–136, 1985.

[56] John Hopcroft and Robert Tarjan. Efficient planarity testing. *Journal of the ACM (JACM)*, 21(4):549–568, 1974.

[57] Yaochun Huang, Xiaofeng Gao, Zhao Zhang, and Weili Wu. A better constant-factor approximation for weighted dominating set in unit disk graph. *Journal of Combinatorial Optimization*, 18(2):179–194, 2009.

[58] Harry B. Hunt III, Madhav V. Marathe, Venkatesh Radhakrishnan, Shankar S. Ravi, Daniel J. Rosenkrantz, and Richard E. Stearns. NC-approximation schemes for NP-and PSPACE-hard problems for geometric graphs. *Journal of algorithms*, 26(2):238–274, 1998.

[59] Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

[60] David S. Johnson. The NP-completeness column: an ongoing guide. *Journal of Algorithms*, 6(3):434–451, 1985.

[61] Frank Kammer, Torsten Tholey, and Heiko Voepel. Approximation algorithms for intersection graphs. In *Proceedings of the 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2010), and 14th International Workshop on Randomization and Computation (RANDOM 2010)*, LNCS 6302, pages 260–273. Springer, 2010.

[62] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2005.

[63] Donghyun Kim, Wei Wang, Xianyue Li, Zhao Zhang, and Weili Wu. A new constant factor approximation for computing 3-connected m-dominating sets in homogeneous wireless networks. In *Proceedings of the 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2010)*, pages 2739–2747. IEEE, 2010.

[64] Ekkehard Köhler and Lalla Mouatadid. A linear time algorithm to compute a maximum weighted independent set on cocomparability graphs. *Information Processing Letters*, 116(6):391–395, 2016.

[65] Erik Jan van Leeuwen. Approximation algorithms for unit disk graphs. In *Proceedings of the 31st International Workshop on Graph-theoretic concepts in computer science (WG 2005)*, LNCS 3787, pages 351–361. Springer, 2005.

[66] Xiang-Yang Li, Peng-Jun Wan, Yu Wang, and Chih-Wei Yi. Fault tolerant deployment and topology control in wireless networks. In *Proceedings of the 4th*

ACM international symposium on Mobile ad hoc networking & computing, pages 117–128. ACM, 2003.

[67] Yingshu Li, My T. Thai, Feng Wang, Chih-Wei Yi, Peng-Jun Wan, and Ding-Zhu Du. On greedy construction of connected dominating sets in wireless networks. *Wireless Communications and Mobile Computing*, 5(8):927–932, 2005.

[68] Yingshu Li, Yiwei Wu, Chunyu Ai, and Raheem Beyah. On the construction of k-connected m-dominating sets in wireless networks. *Journal of combinatorial optimization*, 23(1):118–139, 2012.

[69] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM (JACM)*, 41(5):960–981, 1994.

[70] Madhav V. Marathe, Heinz Breu, Harry B. Hunt III, Shankar S. Ravi, and Daniel J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25(2):59–68, 1995.

[71] Tomomi Matsui. Approximation algorithms for maximum independent set problems and fractional coloring problems on unit disk graphs. In *Proceedings of the Japanese Conference on Discrete and Computational Geometry (JCDCG 2000)*, LNCS 1763, pages 194–200. Springer, 2000.

[72] Manki Min, Hongwei Du, Xiaohua Jia, Christina Xiao Huang, Scott C-H Huang, and Weili Wu. Improving construction for connected dominating set with steiner tree in wireless sensor networks. *Journal of Global Optimization*, 35(1):111–119, 2006.

[73] Rashid Muhammad. Distributed steiner tree algorithm and its application in ad hoc wireless networks. In *Proceedings of the 2006 International Conference on Wireless Networks (ICWN 2006)*, pages 173–178, 2006.

[74] Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010.

[75] Sada Narayanappa and Petr Vojtechovskỳ. An improved approximation factor for the unit disk covering problem. In *Canadian Conference on Computational Geometry*, pages 15–18, 2006.

[76] Tim Nieberg and Johann Hurink. A PTAS for the minimum dominating set problem in unit disk graphs. In *Proceedings of the 3rd International Workshop on Approximation and Online Algorithms (WAOA 2005)*, LNCS 3879, pages 296–306. Springer, 2005.

[77] Tim Nieberg, Johann Hurink, and Walter Kern. A robust PTAS for maximum weight independent sets in unit disk graphs. In *Proceedings of the 30th International Workshop on Graph-theoretic concepts in computer science (WG 2004)*, LNCS 3353, pages 214–221. Springer, 2004.

[78] Tim Nieberg, Johann Hurink, and Walter Kern. Approximation schemes for wireless networks. *ACM Transactions on Algorithms (TALG)*, 4(4):49, 2008.

[79] Mark H. Overmars and Jan van Leeuwen. Worst-case optimal insertion and deletion methods for decomposable searching problems. *Information Processing Letters*, 12(4):168–173, 1981.

[80] B. S. Panda and S. Paul. Connected liar's domination in graphs: complexity and algorithms. *Discrete Mathematics, Algorithms and Applications*, 5(04):1350024 (1–16), 2013.

[81] B. S. Panda and S. Paul. Liar's domination in graphs: Complexity and algorithm. *Discrete Applied Mathematics*, 161(7):1085–1092, 2013.

[82] B. S. Panda and S. Paul. A linear time algorithm for liar's domination problem in proper interval graphs. *Information Processing Letters*, 113(19):815–822, 2013.

[83] B. S. Panda and S. Paul. Hardness results and approximation algorithm for total liar's domination in graphs. *Journal of Combinatorial Optimization*, 27(4):643–662, 2014.

[84] B. S. Panda, S. Paul, and D. Pradhan. Hardness results, approximation and exact algorithms for liar's domination problem in graphs. *Theoretical Computer Science*, 573:26–42, 2015.

[85] Franco P. Preparata and Michael Shamos. *Computational geometry: An introduction*. Springer Science & Business Media, 2012.

[86] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 475–484. ACM, 1997.

[87] Miranda L. Roden and Peter J. Slater. Liar's domination in graphs. *Discrete mathematics*, 309(19):5884–5890, 2009.

[88] Weiping Shang, Frances Yao, Pengjun Wan, and Xiaodong Hu. Algorithms for minimum m-connected k-dominating set problem. In *Proceedings of the 1st International Conference on Combinatorial Optimization and Applications (COCOA 2007)*, LNCS 4616, pages 182–190. Springer, 2007.

[89] Yishuo Shi, Zhao Zhang, Yuchang Mo, and Ding-Zhu Du. Approximation algorithm for minimum weight fault-tolerant virtual backbone in unit disk graphs. *IEEE/ACM Transactions on Networking*, 25(2):925–933, 2017.

[90] Peter J. Slater. Liar's domination. *Networks*, 54(2):70–74, 2009.

[91] Christopher Sterling. *Liar's Domination in Grid Graphs*. PhD thesis, East Tennessee State University, 2012.

[92] My T. Thai, Ravi Tiwari, and Ding-Zhu Du. On construction of virtual backbone in wireless ad hoc networks with unidirectional links. *IEEE Transactions on Mobile Computing*, 7(9):1098–1109, 2008.

[93] My T. Thai, Feng Wang, Dan Liu, Shiwei Zhu, and Ding-Zhu Du. Connected dominating sets in wireless networks with different transmission ranges. *IEEE transactions on mobile computing*, 6(7):721–730, 2007.

[94] Alireza Vahdatpour, Foad Dabiri, Maryam Moazeni, and Majid Sarrafzadeh. Theoretical bound and practical analysis of connected dominating set in ad hoc and sensor networks. In *Proceedings of the 22nd International Symposium on Distributed Computing (DISC 2008)*, LNCS 5218, pages 481–495. Springer, 2008.

[95] Leslie G. Valiant. Universality considerations in VLSI circuits. *IEEE Transactions on Computers*, 100(2):135–140, 1981.

[96] Peng-Jun Wan, Khaled M. Alzoubi, and Ophir Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, volume 3, pages 1597–1604. IEEE, 2002.

[97] Feng Wang, Manki Min, Yingshu Li, and Dingzhu Du. On the construction of stable virtual backbones in mobile ad-hoc networks. In *Proceedings of the 24th International Performance Computing and Communications Conference (IPCCC 2005)*, pages 355–362. IEEE, 2005.

[98] Feng Wang, My T. Thai, and Ding-Zhu Du. On the construction of 2-connected virtual backbone in wireless networks. *IEEE Transactions on wireless communications*, 8(3):1230–1237, 2009.

[99] Yu Wang, WeiZhao Wang, and Xiang-Yang Li. Distributed low-cost backbone formation for wireless ad hoc networks. In *Proceedings of the 6th ACM Interational*

Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2005), pages 2–13. ACM, 2005.

[100] Zhong Wang, Wei Wang, Joon-Mo Kim, Bhavani Thuraisingham, and Weili Wu. PTAS for the minimum weighted dominating set in growth bounded graphs. *Journal of Global Optimization*, 54(3):641–648, 2012.

[101] Weili Wu, Hongwei Du, Xiaohua Jia, Yingshu Li, and Scott C.-H. Huang. Minimum connected dominating sets and maximal independent sets in unit disk graphs. *Theoretical Computer Science*, 352(1-3):1–7, 2006.

[102] Zhao Zhang, Xiaofeng Gao, Weili Wu, and Ding-Zhu Du. PTAS for minimum connected dominating set in unit ball graph. In *Proceedings of the 3rd International Conference on Wireless Algorithms, Systems, and Applications (WASA 2008)*, LNCS 5258, pages 154–161. Springer, 2008.

[103] Feng Zou, Xianyue Li, Suogang Gao, and Weili Wu. Node-weighted steiner tree approximation in unit disk graphs. *Journal of Combinatorial Optimization*, 18(4):342–349, 2009.

[104] Feng Zou, Yuexuan Wang, Xiao-Hua Xu, Xianyue Li, Hongwei Du, Pengjun Wan, and Weili Wu. New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs. *Theoretical Computer Science*, 412(3):198–208, 2011.

Publications from the Contents of the Thesis

Papers published/submitted in international journals

[J1] P. Carmi, Gautm. K. Das, Ramesh. K. Jallu, Subash. C. Nandy, P. R. Prajwal, and Y. Stein, **Minimum Dominating Set Problem for Unit Disks Revisited**, *International Journal of Computational Geometry and Applications (IJCGA)*, Vol. 25, No. 03, pp. 227-244 (2015).

[J2] Ramesh K. Jallu, Prajwal R. Prasad and Gautam K. Das, **Distributed Construction of Connected Dominating set in Unit Disk Graphs**, *Journal of Parallel and distributed computing (JPDC)*, vol. 104 pp. 159-166 (2017).

[J3] Gautam K. Das , Guilherme D. da Fonseca , and Ramesh K. Jallu, **Efficient independent set approximation in unit disk graphs**, `http://fc.isima.fr/~fonseca/udg-is.pdf` (Revised version is submitted to *Discrete Applied Mathematics, 2016).*

[J4] Ramesh K. Jallu, and Gautam K. Das, **Liar's dominating set problem on unit disk graphs**, submitted to *Discrete Applied Mathematics (DAM), 2017.*

Papers published/submitted in conference proceedings

[C1] Ramesh K. Jallu, and Gautam K. Das, **Improved Algorithm for Maximum Independent Set on Unit Disk Graph**, in *Conference on Algorithms and Discrete Applied Mathematics*, Lecture Notes in Computer Science, pp. 212-223, 2016.

[C2] Ramesh K. Jallu, and Gautam K. Das, **Liar's Domination in 2D**, in *Conference on Algorithms and Discrete Applied Mathematics (CALDAM)*, Lecture Notes in Computer Science, pp. 219-229, 2017.

[C3] Ramesh K. Jallu, Sangram K. Jena, and Gautam K. Das, **Liar's Dominating Set in Unit Disk Graphs**, submitted to *The International Computing and Combinatorics Conference (COCOON), 2018.*