Algorithms for Geometric Covering Problems

Thesis submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

by

Manjanna B
(Roll No. 11612315)

Under the Supervision of
Dr. Gautam Kumar Das

to the

Department of Mathematics
Indian Institute of Technology Guwahati
Guwahati - 781039, India

July 2016

CERTIFICATE

This is to certify that this thesis entitled "**Algorithms for Geometric Covering Problems**" being submitted by Mr. Manjanna B 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: July 23, 2016

Dr. Gautam Kumar Das
Department of Mathematics
Indian Institute of Technology Guwahati
Guwahati-781039, Assam, India

ACKNOWLEDGEMENTS

Undertaking this PhD at Indian Institute of Technology Guwahati, has been an unforgettable learning and knowledge gaining experience for me. It would not have been possible to do without the support and guidance that I received from many people. It has been a great privilege to spend these four years in the Department of Mathematics at Indian Institute of Technology Guwahati, and its members will always remain dear to me.

First and foremost, I want to express my deepest appreciation and thanks to my supervisor Dr. Gautam Kumar Das, who agreed to take me as his student and introduced me to the field of geometric covering and approximation algorithms. He patiently provided the vision, encouragement and advise necessary for me to proceed through the doctoral program and complete my thesis work. I have immensely benefited from his careful reading of my works, constructive feedbacks, ongoing encouragement and stimulating conversation with him. Besides all these, whenever I faced difficulties, he oriented me to go ahead with his patient guidance, utmost care and concern for the whole of my doctoral journey. I am grateful, and thank him for all the guidance he provided me, both in academic and personal contexts. I also want to thank my doctoral committee chairman, Prof. S. V. Rao, and doctoral committee members, Dr. Partha Sarathi Mondal and Dr. Sushanta Karmakar, for their valuable comments and suggestions during this thesis work. I acknowledge the Ministry of Human Resource Development, Government of India, for providing me financial support for pursuing PhD at IIT Guwahati. I also thank the Head of Department of Mathematics, IIT Guwahati, for providing me all the necessary facility for carrying out my research at IITG. I thank all the staff members of the Department of Mathematics, IIT Guwahati for their assistance in official and technical matters.

Many details of this thesis are improved upon or corrected based on the review, comments and suggestions by anonymous reviewers. I am also thankful to them. I would also like to thank Prof. Subhas C. Nandy for his collaborative work with us and being the co-author of a publication from this thesis. I enjoyed discussing problems and algorithms, and writing Matlab code with Ramesh. I am thankful to him for all his help. I thank Rashmisnata Acharyya for being the co-author of our paper. My sincere thanks also goes to Dr. Ramanathan M., for offering me the summer internship opportunity in their group at IIT Madras.

I wish to thank all my friends, fellow research scholars, lab mates, cubical mates etc. at IIT Guwahati for their company during my stay in the IIT campus. I thank Barun, Shibsankar, Ramesh, Anand, Debasish, Subramanyam, Tamal, Manish, Swarup, Dinesh, Punit, Jayant, Mrityunjay, Amit, Ashok sir, Ramesh sir, to name a few, for their support and useful advices during my days in IIT Guwahati. I am grateful to all others who have helped me in various ways directly or indirectly while doing this thesis work at IIT Guwahati.

I would also like to acknowledge all my friends from my school days, with whom I spent best of my student life. I wish to thank my friends Ankit and Malay, who always encouraged me to pursue an advanced degree, and helped me and gave me useful advices in various ways since the days before my MTech.

I must thank my parents for all their effort they devoted to getting me an education. Without them, I may never have gotten to where I am today. I also want to thank to my in-laws for their unconditional support. At the end I would like to express appreciation to my beloved wife Nalina who has taken care of me and my little son Vipin all the time, shared in my stress and joy, and given me support every day. I thank her for being understanding and cooperative during times when I was not able to spend quality time with the family.

Place: IIT Guwahati

Date: 23-7-2016 Manjanna B

ABSTRACT

Motivated by the applications in facility location, VLSI design, image processing and motion planning, geometric covering problems have been studied extensively in the literature. In this thesis various geometric covering problems such as covering points with disks, and squares, covering rectangular regions and convex polygonal regions with disks are considered. The problems are investigated by proposing approximation, parameterized and heuristic algorithms.

The Discrete Unit Disk Cover (DUDC) problem is one of the well known geometric covering problems. In the DUDC problem, the input consists of a set $\mathcal{P}$ of n points and a set $\mathcal{D}$ of m unit disks in $I\!R^2$. The objective is to (i) check whether the union of all the disks in $\mathcal{D}$ cover all the points in $\mathcal{P}$, and (ii) if yes, then find a minimum size subset $\mathcal{D}^* \subseteq \mathcal{D}$, whose union covers all points in $\mathcal{P}$. The DUDC problem is a NP-complete problem, so there does not exist any polynomial time algorithm that solves the problem optimally unless $P = NP$. In order to approximate the DUDC problem, several restricted variants of the DUDC problem have been examined by different researchers. We improve the approximation factor of one such restricted DUDC problem, which helps to improve the approximation factor of the DUDC problem.

We consider the Line-Separable Discrete Unit Disk Cover (LSDUDC) problem as a restricted version of the DUDC problem. In the LSDUDC problem, the plane is divided into two half planes ℓ^+ and ℓ^- defined by a line ℓ, all the points in $\mathcal{P}$ are in ℓ^- and the centers of the disks in $\mathcal{D}$ are in $\ell^+ \cup \ell^-$ such that each point in $\mathcal{P}$ is covered by at least one disk centered in ℓ^+. We provide a Polynomial Time Approximation Scheme (PTAS) for the LSDUDC problem. Another restricted DUDC problem is called the Within-Strip Discrete Unit Disk Cover (WSDUDC) problem. In the WSDUDC problem, all points in $\mathcal{P}$ and the centers of all disks in $\mathcal{D}$ are inside a horizontal strip of height $\frac{1}{\sqrt{2}}$. The current best known algorithm for the WSDUDC problem is a 3-approximation algorithm [33]. We propose a $(9+\epsilon)$-approximation algorithm for the DUDC problem $(0 < \epsilon \leq 6)$ using the proposed PTAS result for the LSDUDC problem, and a 3-approximation algorithm for the WSDUDC problem. The running time of the proposed algorithm for the DUDC problem is $O(m^{3(1+\frac{6}{\epsilon})} n \log n)$.

We also consider another unit disk cover problem, namely the Rectangular Region

Cover (RRC) problem. In the RRC problem, given rectangular region $\mathcal{R}$ and a set $\mathcal{D}$ of m unit disks in $\mathbb{R}^2$, the objective is (i) to check whether the union of all the disks in $\mathcal{D}$ covers the entire region $\mathcal{R}$, and (ii) if $\mathcal{D}$ covers $\mathcal{R}$, then determine the minimum cardinality set $\mathcal{D}^* \subseteq \mathcal{D}$ such that the region $\mathcal{R}$ is contained in the union of all disks in $\mathcal{D}^*$. By mapping every instance of the RRC problem to an instance of the DUDC problem, we provide a $(9+\epsilon)$-approximation algorithm for the RRC problem using our algorithm for the DUDC problem, where $n = O(m^2)$. We also consider the RRC problem in a reduced radius setup. The RRC problem in reduce radius setup has important application in wireless sensor networks, where coverage remains stable under small perturbations of sensing ranges and positions of sensors. We obtain a PTAS for the RRC problem in reduce radius setup using the shifting strategy of Hochbaum and Maass [50].

Discrete Unit Square Cover (DUSC) problem is an L_∞ metric variant (or an L_1 metric variant) of the DUDC problem. In the DUSC problem, given a set $\mathcal{P}$ of n points and a set $\mathcal{S}$ of m axis-aligned unit squares (unit side length) in $\mathbb{R}^2$, the objective is (i) to check whether the union of all the squares in $\mathcal{S}$ covers all the points in $\mathcal{P}$, and (ii) if $\mathcal{S}$ covers $\mathcal{P}$, then determine the minimum cardinality set $\mathcal{S}^* \subseteq \mathcal{S}$ such that each point in $\mathcal{P}$ is covered by at least one square in $\mathcal{S}^*$. We consider a restricted version of the DUSC problem, namely Strip Square Cover (SSC) problem. In the SSC problem, all n points of $\mathcal{P}$ lie within a horizontal strip of unit height. We first propose an $(1+\frac{2}{k-2})$-approximation algorithm for the SSC problem in $O(km^k n)$ time, then using the result for SSC problem, we propose a $(2+\frac{4}{k-2})$-approximation algorithm for the DUSC problem, where $k(>2)$ is an integer parameter that defines a trade-off between the running time and the approximation factor of the algorithm. The running time of the proposed algorithm for the DUSC problem is $O(km^k n)$. We also outline a 2-approximation algorithm for the DUSC problem, which runs in $O(m^4 n + n \log n)$ time.

Unlike in the above unit disk cover or unit square cover problems, in some of the geometric covering problems the number of disks is fixed and the radius of disks is not fixed. One such geometric covering problem is called the k-center problem. In the (geometric) k-center problem, given a set $\mathcal{P}$ of n points (clients), the objective is to place k congruent disks of smallest radius (facilities) such that the union of the k congruent disks covers $\mathcal{P}$. The centers of the k congruent disks can be arbitrary points in the plane or must be chosen from a given discrete set $\mathcal{Q}$ ($\subseteq \mathcal{P}$) of m ($\leq n$) points, representing candidate locations to establish facilities. In the latter case, if $\mathcal{P} = \mathcal{Q}$, then the k-center problem is called the discrete k-center problem and if $\mathcal{P} \neq \mathcal{Q}$, then it is a generalized discrete k-center problem or k-supplier problem. In the former case, sometimes the centers of the k disks are constrained to lie on the boundary of a convex polygon P. This variation of the k-center problem is called Constrained Convex Polygon Cover (CCPC) problem. For the CCPC problem, we propose $(1 + \frac{7}{k} + \frac{7\epsilon}{k} + \epsilon)$-approximation algorithm for an $\epsilon > 0$ and an integer $k \geq 7$ such that the union of the k congruent disks covers P. The running time of the proposed algorithm for CCPC problem is $O(n(n+k)(|\log r_{opt}| + \log\lceil\frac{1}{\epsilon}\rceil))$, where n is the number of vertices of the convex polygon P, r_{opt} is the minimum radius of k congruent disks. For the k-supplier problem in $I\!R^2$, we propose a fixed parameter tractable (FPT) 2-approximation algorithm, where k is the parameter. The running time of the proposed FPT 2-approximation algorithm is $O(6^k(n + m)(\log n + \log m))$. We can generalize the technique used for developing FPT 2-approximation algorithm to develop a FPT $(1+\epsilon)$-approximation algorithm for the k-supplier problem in $I\!R^d$, where d is a positive integer and $\epsilon > 0$ is an arbitrary number. The running time of the proposed $(1 + \epsilon)$-approximation algorithm is $O(\epsilon^{-dk}(m + n)\log(mn))$. We also present a heuristic algorithm based on nearest point Voronoi diagram for the Euclidean k-supplier problem in $I\!R^2$ and experimentally show that it performs very well.

Contents

1 Introduction **1**

1.1 Scope of the Thesis 4

1.2 Organization of the Thesis 4

2 Literature Review **7**

3 Discrete Unit Disk Cover Problem **19**

3.1 Previous Approximation Algorithms for DUDC Problem 21

3.2 PTAS for the LSDUDC Problem 23

3.2.1 $(9 + \epsilon)$-Approximation Algorithm for DUDC Problem 32

3.3 Approximation Algorithms for the RRC Problem 33

3.3.1 RRC Problem in Reduce Radius Setup 34

3.4 Conclusion 37

4 Discrete Unit Square Cover Problem **39**

4.1 Testing Feasibility of the Discrete Unit Square Cover Problem 40

4.2 Strip Square Cover Problem 41

4.2.1 Terminology 42

4.2.2 Preliminaries 42

4.2.3 Approximation Algorithm for Strip Square Cover Problem 44

4.3 Approximation Algorithm for the Discrete Unit Square Cover Problem . 46

4.4 2-Approximation Algorithm for the DUSC Problem 48

4.4.1 Algorithm for the Strip Square Cover (SSC) Problem 48

4.5 Conclusion 53

5 Constrained k-Center Problem on a Convex Polygon **54**
5.1 Decision Version of CCPC Problem 55
5.1.1 Preprocess 55
5.1.2 Approximation Algorithm for the Decision Version of the Problem 56
5.2 Constrained Convex Polygon Cover Problem 67
5.3 Conclusion 70

6 The Euclidean k-Supplier Problem **71**
6.1 FPT 2-Approximation Algorithm 72
6.1.1 Terminologies 72
6.1.2 Approximation Algorithm 73
6.2 Heuristic Algorithm for the k-Supplier Problem 78
6.2.1 Experimental Results 83
6.3 Conclusion 83

7 Conclusions and Future Works **84**

List of Figures

3.1 Lower region, left intersection and lower boundary disks 24
3.2 Proof of Claim 3.2.1 25
3.3 Proof of Lemma 3.2.4 25
3.4 Proof of Lemma 3.2.5 26
3.5 Proof of Lemma 3.2.6 28
3.6 Proof of Lemma 3.2.6 (mirror cases) 29
3.7 Proof of Lemma 3.2.6 (mirror cases) 30
3.8 Definition of different regions 35

4.1 An instance of the SSC problem 42
4.2 Illustration of two squares in $\mathcal{S}^{\ell_1}$ intersecting line L and strip $\mathcal{H} = [\ell_1, \ell_2]$ 43
4.3 Illustration of two squares in $\mathcal{S}^{\ell_2}$ intersecting line L and strip $\mathcal{H} = [\ell_1, \ell_2]$ 44
4.4 Illustration of $\mathcal{S}^i \cap \mathcal{S}^{i+2} = \emptyset$ 46
4.5 (a) Lower envelope, and (b) Upper envelope 50
4.6 *Successor* $T' = (u', v', x')$ of *Triplet* $T = (u, v, x)$ 51

5.1 Constrained placement of disks 56
5.2 Non-constrained placement 61
5.3 Blue colored disk centered by non-constrained placement 62
5.4 Proof of Observation 5.1.7 63
5.5 Proof of Lemmata 5.1.8 and 5.1.9 64
5.6 Proof of Theorem 5.1.13 68

6.1 Partition of the disk $\Delta(p, r)$ into six equal sectors $\Delta^1, \Delta^2, \ldots, \Delta^6$ 74

List of Tables

2.1 Approximation algorithms for the DUDC problem 8
2.2 Approximation algorithms for the CUDC problem 11

6.1 Radii of disks centered by different algorithms and their execution time . 82

Abbreviations and Acronyms

- ℓ^+ and ℓ^- - half-planes above and below horizontal line ℓ respectively.
- L^- and L^+ - half-planes to the left and right of the vertical line L respectively.
- $\theta(d)$ and $\alpha(d)$ - the boundary arc and center of the disk d respectively.
- $\overline{a,b}$ - line segment joining the points a and b.
- $(x(s),\, y(s))$ - coordinates of the center of square s.
- $(x(p), y(p))$ - coordinates of the point p.
- $a \leftarrow b$ - variable a gets the value of b.
- $dist(p', p'')$, $\delta(p', p'')$ - the Euclidean distance between the points p' and p''.
- $(x'_i,\, y'_i)$ - coordinates of the center of disk d'_i.
- P - convex polygon unless otherwise specified.
- ∂P - boundary line of the polygon P.
- ∂d_i - boundary arc of the disk d_i.
- U_c and L_c - upper chain and lower chain of the polygon P respectively.
- $\alpha(i)$ - disk d'_s such that $x_{i+2} \leq x'_s \leq x_i$ and centered on the same chain as disks d_i and d_{i+2} or $x_{i+3} \leq x'_s \leq x_{i+1}$ and centered on the same chain as disks d_{i+1} and d_{i+3}, where $1 \leq s \leq k$.
- $\Delta(a, r)$ - region covered by the disk of radius r centered at point a.

Chapter 1

Introduction

Geometric covering is a well studied topic in computational geometry. Various types of covering problems include covering one type of geometric object with a minimum number of some other type of geometric object. The wide range of geometric objects includes points, lines, disks, squares and rectangles. In the geometric disk cover problems, the objective is to cover all points in a given set $\mathcal{P}$ of points in the Euclidean plane or all points within a given continuous region such as rectangular region $\mathcal{R}$ and convex polygon P, using the minimum number of disks. If the disks covering points or continuous region are centered at points in a set of points, then we call this type of geometric disk cover problems as the *discrete disk cover* problem. If the disks covering points or continuous region are centered at arbitrary points, then we call this type of geometric disk cover problem a *continuous disk cover* problem. A version of the *discrete disk cover* problem in which all the covering disks have unit radius is called the *discrete unit disk cover* (DUDC) problem. In the *unit disk cover* (UDC) problem, we consider two variations of the problem, namely the *discrete unit disk cover* (DUDC) problem and the *rectangular region cover* (RRC) problem. In the DUDC problem, given a set $\mathcal{P}$ of n points and a set $\mathcal{D}$ of m unit disks in the plane, we wish (i) to check whether the union of the disks in $\mathcal{D}$ covers all the points in $\mathcal{P}$, and (ii) if yes, then determine the minimum cardinality set $\mathcal{D}^* \subseteq \mathcal{D}$ such that $\mathcal{P} \subseteq \bigcup_{d \in \mathcal{D}^*} d$. In the *rectangular region cover* (RRC) problem, given a rectangular region $\mathcal{R}$ and a set $\mathcal{D}$ of m unit disks in the plane, the objective is (i) to check whether the union of the disks in $\mathcal{D}$ covers the entire region $\mathcal{R}$, and (ii) if $\mathcal{D}$ covers $\mathcal{R}$, then determine the minimum cardinality set $\mathcal{D}^{**} \subseteq \mathcal{D}$ such that $\mathcal{R} \subseteq \bigcup_{d \in \mathcal{D}^{**}} d$. The DUDC and the RRC problems are geometric versions of the general set cover problem, which is known to be NP-complete [38]. The general set cover problem is not approximable within a factor of $c \log n$, for some constant c, where n is the size of the input [72]. However, the DUDC, and the RRC problems admit constant factor approximation results. These two problems have been studied extensively due to their wide applications in wireless networks [15, 32, 80].

Consider the facility location problems for which the Euclidean distance between clients and facilities cannot exceed a fixed distance r, and clients and candidate facility locations are represented by discrete sets of points, for example, positioning emergency service centers (i.e. fire stations) from a set of candidate sites so that all points of interest (houses, etc.) are within a predefined maximum distance of the service centers. Other applications are selecting locations for wireless servers from a set of candidate locations to cover a set of wireless clients, selecting a set of weather radar antennae to cover a set of cities, positioning a fleet of water bombers at airports such that every active forest fire is within a given maximum distance of a water bomber, selecting locations for anti-ballistic defenses from a set of candidate locations to cover strategic sites. These facility location problems can be treated as a *discrete disk cover* problem where the centers of disks of fixed radius r represent the facilities and points represent the clients. The *discrete disk cover* problem with the disks of fixed radius r can be mapped to a *discrete unit disk cover* problem by scaling down the original problem with a factor of distance r. We can obtain the solution for the original problem by scaling up the solution computed for the DUDC problem. Thus, the model of *unit disk cover* problems applies naturally to several facility location problems for which the Euclidean distance between clients and facilities cannot exceed a given radius, and clients and candidate facility locations are represented by discrete sets of points.

In the DUSC problem, given a set $\mathcal{P}$ of n points and a set $\mathcal{S}$ of m axis-aligned unit squares (unit side length) in $\mathbb{R}^2$, we wish (i) to check whether the union of the squares in $\mathcal{S}$ covers all the points in $\mathcal{P}$, and (ii) if so, then determine the minimum cardinality set $\mathcal{S}^* \subseteq \mathcal{S}$ such that $\mathcal{P} \subseteq \bigcup_{s \in \mathcal{S}^*} s$. The DUSC problem has several applications in image processing [79]. Note that the DUSC problem is an L_∞ variant (or an L_1 variant) of the *discrete unit disk cover* (DUDC) problem as follows: for a point p, we define the area A_p such that the distance between any point q in A_p from the point p is less than or equal to one unit in the L_1 metric[1]. Note that the shape of A_p is a square with side length $\sqrt{2}$ and tilted with angle $\frac{\pi}{4}$ with x-axis. Therefore, the DUDC problem in L_1 metric is equivalent to the *discrete square cover* problem after rotating axes by an angle $\frac{\pi}{4}$. Also the DUDC problem in L_∞ metric is equivalent to the *discrete square cover* problem. Like the DUDC problem, the DUSC problem can also be formulated as a set cover problem. The DUSC problem can be treated as a geometric version of the set cover problem. Thus, any algorithm for the set cover problem can be used to solve the DUSC problem. The DUSC problem is also a NP-complete problem [35]. The DUSC problem, however, admits constant factor approximation results.

In this thesis we consider another interesting covering problem known as the (geometric) *k-center* problem. In the *k-center* problem, a set of clients (e.g. mobile users, houses) are distributed in $\mathbb{R}^2$. The objective is to choose k locations for facilities (e.g. base stations for mobile networks, post office, warning sirens) so that each client can get service from at least one facility within a minimum distance. From the geometric point of view, the *k-center* problem is to cover all n points (clients) with the union of k congruent disks (facilities) of radius as small as possible. In the *k-center* problem, if the centers of k facilities are chosen from a given set of points, then the *k-center* problem is called the *discrete k-center* problem. In this thesis we consider a generalization of the *discrete k-center* problem, which is known as the *k-supplier* problem in the literature [69]. Here, a set $\mathcal{P}$ of n clients (customer sites) and a set $\mathcal{Q}$ of m facilities (supplier sites) are given. The objective is to open a set $Q_{opt} \subseteq \mathcal{Q}$ of k facilities such that the maximum distance of a client to its nearest facility from Q_{opt} is minimized. The *k-supplier* problem has numerous applications including facility location (e.g. placing k hospitals at some specified locations such that the maximum distance from any house to its nearest hospital is minimized), information retrieval and data mining.

Sometimes, we may need to restrict these facilities to be located only on the boundary of the given region (containing all clients), for example base station placement in a forbidden region (such as a big lake) [24, 71]. In the context of the application of k-

[1]unless otherwise specified, we assume that all the problems considered in this thesis are in L_2 metric

center problems, the next problem considered in this thesis is called the *restricted* or *constrained k-center* problem. The formal definition of the *constrained k-center* problem is as follows: given a convex polygon P and an integer k, the objective is to cover the entire region of P with k congruent disks of minimum radius and centered on the boundary of P.

1.1 Scope of the Thesis

In this thesis we consider different geometric covering problems involving various geometric objects. Most of the problems that we consider in this thesis are NP hard and the hardness of the remaining problems are unknown. Hence, our focus is to design efficient approximation algorithms for different covering problems. In this thesis we consider the following geometric covering problems: *line separable discrete unit disk cover* (LSDUDC) problem, *discrete unit disk cover* (DUDC) problem, *discrete unit square cover* (DUSC) problem, *rectangular region cover* (RRC) problem, RRC problem in reduced radius setup, *constrained k-center* problem on a convex polygon and *Euclidean k-supplier* problem in the plane. For both the LSDUDC problem, and the RRC problem in reduced radius setup, we have developed a polynomial time approximation scheme (PTAS). For the DUDC problem, RRC problem, and *constrained k-center* problem on a convex polygon, we have proposed constant factor approximation algorithms. For the *Euclidean k-supplier* problem, we have presented fixed parameter tractable (FPT) constant factor approximation algorithms. For the *Euclidean k-supplier* problem in $\mathbb{R}^2$, we have also developed a heuristic algorithm and studied its behavior theoretically as well as experimentally.

1.2 Organization of the Thesis

Chapter 2: Literature Review. In this chapter we discuss previous work on the problems related to this thesis and compare our work with existing research.

Chapter 3: Discrete Unit Disk Cover Problem. In this chapter we begin with an outline of existing algorithms for DUDC and related problems. We then discuss our proposed PTAS ($(1+\mu)$-approximation algorithm and $\mu > 0$) for a restricted DUDC problem, namely the *line separable discrete unit disk cover* (LSDUDC) problem. In the LSDUDC problem, the plane being divided into two half planes ℓ^+ and ℓ^- defined by a line ℓ, all the points in $\mathcal{P}$ are in ℓ^- and the centers of the disks in $\mathcal{D}$ are in $\ell^+ \cup \ell^-$ such that each point in $\mathcal{P}$ is covered by the union of the disks centered in ℓ^+. Using our proposed PTAS for the LSDUDC problem, we present a $(9+\epsilon)$-approximation algorithm for the DUDC problem, where $\epsilon > 0$. The running time of the algorithm is $O(m^{3(1+\frac{6}{\epsilon})} n \log n)$, where m is the number of unit disks and n is the number of points. We then propose a $(9+\epsilon)$-approximation algorithm for the RRC problem. The running time of the algorithm for the RRC problem is $O(m^{5+\frac{18}{\epsilon}} \log m)$. We also consider the RRC problem in a different setup called reduce radius setup, which has important application in wireless sensor networks. For the RRC problem in reduced radius setup, we propose an $(1+1/l)^2$-approximation algorithm (PTAS), where l is a positive integer. The running time of the PTAS is $O(ql^2 2^{\lceil \frac{4l^2}{\nu^2} + \frac{8l+4}{\nu} \rceil})$, where q is the minimum number of squares of size $2l \times 2l$ covering rectangular region $\mathcal{R}$.

Chapter 4: Discrete Unit Square Cover Problem. In this chapter we first describe a procedure to check the feasibility of a given instance of DUSC problem. We then define a subproblem of the DUSC problem in which all the points are lying within a horizontal strip of unit height. We call this restricted DUSC as *strip square cover* (SSC) problem. We propose an $(1+\frac{2}{k-2})$-approximation algorithm for the SSC problem, where $k(>2)$ is an integer parameter that defines a trade-off between the running time and the approximation factor of the algorithm. The running time of the algorithm is $O(km^k n)$. Using this algorithm for the SSC problem, we propose a $(2+\frac{4}{k-2})$-approximation algorithm for the DUSC problem. The running time of our proposed algorithm for the DUSC problem is $O(km^k n)$. For the DUSC problem, we also propose a 2-approximation algorithm, which runs in $O(m^4 n + n \log n)$ time.

Chapter 5: Constrained k-Center Problem on a Convex Polygon. In this chapter we first define a decision version of the *constrained convex polygon cover* (CCPC) problem. We then propose an $(1+\frac{7}{k})$-factor approximation algorithm for the decision version of the CCPC problem ($k \geq 7$), which runs in $O(n^2+nk)$ time. Using this algorithm for the decision version, we propose an $(1+\frac{7}{k}+\frac{7\epsilon}{k}+\epsilon)$-approximation algorithm for the CCPC problem for $\epsilon > 0$. The running time of the proposed CCPC algorithm is $O(n(n+k)(|\log r_{opt}|+\log\lceil\frac{1}{\epsilon}\rceil))$, where n is the number of vertices in the convex polygon P, r_{opt} is the optimum radius of k congruent disks such that the union of the k congruent disks covers P.

Chapter 6: The Euclidean k-Supplier Problem. In this chapter we initially consider the Euclidean k-supplier problem in $\mathbb{R}^2$. We then propose a fixed-parameter tractable (FPT) algorithm for this problem that produces a 2-approximation result. The worst case running time of this FPT approximation algorithm is $O(6^k(n+m)\log(mn))$, where k is the FPT parameter. We can generalize the FPT 2-approximation algorithm to develop a FPT $(1+\epsilon)$-approximation algorithm for k-supplier problem in $\mathbb{R}^d$, where d is a positive integer and $\epsilon > 0$ is an arbitrary number. The running time of the proposed $(1+\epsilon)$-approximation algorithm is $O(\epsilon^{-dk}(m+n)\log(mn))$. We also propose a heuristic algorithm for the Euclidean k-supplier problem in $\mathbb{R}^2$ and experimentally show that the proposed heuristic performs very well for randomly generated instances.

Chapter 7: Conclusion and Future Works. In this chapter we make the concluding remarks and identify some open problems which can be considered in future research.

Chapter 2

Literature Review

Geometric covering is a well-studied problem in the literature. First we consider the *discrete unit disk cover* (DUDC) problem. In the DUDC problem, a set $\mathcal{P}$ of n points and a set $\mathcal{D}$ of m unit disks centered at the m points in $\mathcal{Q}$ are given, the objective is (i) to check whether the union of all the disks in $\mathcal{D}$ cover all the points in $\mathcal{P}$, and (ii) if yes, then choose the minimum cardinality set $\mathcal{D}^* \subseteq \mathcal{D}$ such that each point of $\mathcal{P}$ is covered by the union of the disks in $\mathcal{D}^*$. The DUDC problem has a long history in the literature. It is a NP-complete problem [38]. The first constant factor approximation algorithm was proposed by Brönnimann and Goodrich [8] using the concept of epsilon net. After that many authors proposed constant factor approximation algorithms for the DUDC problem [6, 15, 19, 20, 66, 70]. Using local search, Mustafa and Ray [66] proposed a PTAS for the DUDC problem. The time complexity of their PTAS is $O(m^{2(\frac{8\sqrt{2}}{\epsilon})^2+1}n)$ for $0 < \epsilon \leq 2$. Thus for $\epsilon = 2$, we have a 3-approximation result in $O(m^{65}n)$ time, which is not practical. This led to further research on the DUDC problem for finding constant factor approximation algorithm with reasonable running time. Das et al. [23] proposed an 18-approximation algorithm. The running time of their algorithm is $O(mn+n\log n+m\log m)$. Recently, Fraser and López-Ortiz [33] proposed a 15-approximation algorithm for the DUDC problem, which runs in $O(m^6n)$ time. In this thesis, we propose a $(9+\epsilon)$-approximation algorithm, which runs in $O(m^{3(1+\frac{6}{\epsilon})}n\log n)$ time for $0 < \epsilon \leq 6$. The detailed summary of results on the DUDC problem is given in Table 2.1.

In solving the DUDC problem, some authors consider a restricted version of the

Approximation factor	Running time	Reference
108	polynomial time	Călinescu et al., 2004
72	polynomial time	Narayanappa & Voytechovsky, 2006
38	$O(m^2n^4)$	Carmi et al., 2007
22	$O(m^2n^4)$	Claude et al., 2010
18	$O(mn + n\log n + m\log m)$	Das et al., 2012
15	$O(m^6n)$	Fraser & López-Ortiz, 2012
$(9+\epsilon)$ for $0 < \epsilon \leq 6$	$O(m^{3(1+\frac{6}{\epsilon})} n \log n)$	This thesis
$(1+\epsilon)$ for $0 < \epsilon \leq 2$	$O(m^{2(\frac{8\sqrt{2}}{\epsilon})^2+1} n)$	Mustafa and Ray, 2009

Table 2.1: Approximation algorithms for the DUDC problem

DUDC problem, which is known as *line-separable discrete unit disk cover* (LSDUDC) problem in the literature [23]. In this problem, the plane being divided into two half-planes ℓ^+ and ℓ^- defined by a line ℓ, all the points in $\mathcal{P}$ are in ℓ^- and the centers of disks in $\mathcal{D}$ are in $\ell^+ \cup \ell^-$ such that each point in $\mathcal{P}$ is covered by at least one disk centered in ℓ^+. If the centers of all the disks in $\mathcal{D}$ are in ℓ^+, then this setting of DUDC problem is known as *restricted line-separable discrete unit disk cover* (RLSDUDC) problem. Carmi et al. [20] described a 4-approximation algorithm for the LSDUDC problem. Later, Claude et al. [15] proposed a 2-approximation algorithm for the LSDUDC problem and polynomial time algorithm for the RLSDUDC problem. In this thesis we have proposed a $(1+\epsilon)$-approximation algorithm for the LSDUDC problem for $0 < \epsilon \leq 1$. Another restricted version of the DUDC problem is the *within strip discrete unit disk cover* (WSDUDC) problem, where all the points in $\mathcal{P}$ and the centers of all the disks in $\mathcal{D}$ lie inside a strip of height δ. Das et al. [23] proposed a 6-approximation algorithm for $\delta = 1/\sqrt{2}$. Later, Fraser and López-Ortiz [33] proposed a $3\lceil 1/\sqrt{1-\delta^2}\rceil$-approximation result for $0 \leq \delta < 1$ and proved that WSDUDC is NP-complete. They also proposed a 3-approximation (resp. 4-approximation) algorithm for $\delta \leq 4/5$ (resp. $\delta \leq 2\sqrt{2}/3$).

Das et al. [22] studied another restricted version of the DUDC problem. In this version of the DUDC problem, the centers of all the disks in $\mathcal{D}$ are within a unit disk and all the points in $\mathcal{P}$ are outside of that unit disk. They proposed a 2-approximation algorithm for this restricted version of the DUDC problem, which runs in $O((m+n)^2)$ time. Another well-studied restricted version of the DUDC problem ($\mathcal{P} = \mathcal{Q}$) is called the *geometric minimum dominating set* (GMDS) problem. The GMDS is defined as follows: given a set $\mathcal{P}$ of n points in the plane, find a minimum cardinality set $\mathcal{P}' \subseteq \mathcal{P}$ such that every point $p \in \mathcal{P}$ lies in a unit disk centered at some point of $\mathcal{P}'$. In other words, given a set $\mathcal{P}$ of points, let $G = (V, E)$ be a unit disk graph defined as follows: the vertex set V corresponds to the point set $\mathcal{P}$ and each edge $e = (v_i, v_j) \in E$ if and only if the unit disks centered at v_i and v_j intersect. Now, the objective is to find a minimum cardinality dominating set in the graph G. Since the GMDS problem is a restricted version of the DUDC problem, all the above algorithms given for the DUDC problem are also applicable for the GMDS problem. It is known that the GMDS problem is NP-hard [14]. An $(1+\mu)$-approximation algorithm (PTAS) for $0 < \mu \leq 1$ is given by Nieberg and Hurink [67]. The PTAS of Nieberg and Hurink [67] accepts any undirected graph as input and returns a dominating set of desired bound (i.e. depending on value of μ) if the input graph satisfies the characterizatin of unit disk graph, otherwise a certificate showing that the input graph is not a unit disk graph. For $\mu = 1$ their algorithm becomes a 2-approximation algorithm, which runs in $O(n^{81})$ time [21]. Gibson and Pirwani [41] also proposed a $(1 + \epsilon)$-approximation algorithm (PTAS) for an even more generalized version of the GMDS problem, namely the minmum dominating set problem of arbitrary size disk graph. The running time of their PTAS is $n^{O(\frac{1}{\epsilon^2})}$. Marathe et al. [64] presented a 5-approximation algorithm, which runs in $O(n^2)$ time. In the minimum weight dominating set (MWDS) problem, each node of the graph has a positive weight and the objective is to find a minimum weight dominating set in the graph. Ambühl et al. [6] proposed 72-approximation algorithm for the MWDS problem. Later, the approximation factor was improved to $6+\epsilon$, $5+\epsilon$ and $4+\epsilon$ by Huang et al. [48], Dai and Yu [27], and Zou et al. [81], respectively. Initially, they developed a δ-approximation algorithm ($\delta = 6, 5, 4$) for a subproblem of the MWDS problem, and using this result they proposed $(\delta + \epsilon)$-approximation algorithm for the original MWDS problem. Their algorithms run in $O(\alpha(n)\beta(n))$ time, where $O(\alpha(n))$ is the running time of the algorithm for the subproblem and $O(\beta(n)) = O(n^{4(\lceil\frac{84}{\epsilon}\rceil)^2})$ is the number of times the subproblem needs to be invoked to solve the original MWDS problem. For $\epsilon = 1$ their algorithm produces $(\delta + 1)$-approximation result, but the time complexity becomes very huge as $\alpha(n)\beta(n)$ is a very high degree polynomial function in n. Carmi et al. [18] considered the GMDS problem on an arbitrary size disk graph and proposed a 5-approximation algorithm. Using the local improvement technique, Fonseca et al. [31] proposed $\frac{44}{9}$-approximation algorithm for the GMDS problem. The running time of this algorithm is $O(n \log n)$. In the same paper they proposed a $\frac{43}{9}$-approximation algorithm for the variation of the GMDS problem in which instead of the coordinates of disk centers the adjacency list representation of the graph is given. The running time of this algorithm is $O(n^2 m)$. De et al. [21] presented a δ-approximation algorithms for $\delta = 12, 4$ and 3 with time complexites $O(n \log n)$, $O(n^8 \log n)$ and $O(n^{15} \log n)$ respectively for the GMDS problem. Recently, Carmi et al. [16] proposed a series of approximation algorithms for the GMDS problem. They first proposed a very simple 5-approximation algorithm. The running time of the algorithm in $O(n \log k)$ time, where k is the output size. They then improved the time complexity of De et al. [21]'s 4-approximation algorithm for the GMDS problem to $O(n^6 \log n)$. They showed that a minor modification of this algorithm produces a $\frac{14}{3}$-approximation algorithm, which runs in $O(n^5 \log n)$ time. They proposed a 3-approximation algorithm, which runs in $O(n^{11} \log n)$ time for the GMDS problem. They also proposed a $\frac{45}{13}$-approximation algorithm, which runs in $O(n^{10} \log n)$ time, for the GMDS problem. Finally, they developed a novel shifting strategy and using that strategy they presented $\frac{5}{2}$-approximation algorithm and PTAS for the GMDS problem [16].

In the *continuous unit disk cover* (CUDC) problem, a set $\mathcal{P}$ of n points is given in the Euclidean plane, and the objective is to compute a minimum cardinality set OPT of unit disks such that each point in $\mathcal{P}$ is covered by at least one disk in OPT. Fowler et al. [34] proved that the CUDC problem is NP-hard. For points in $I\!R^d$ and any positive integer $l \geq 1$, Hochbaum and Maass [50] proposed a $(1 + \frac{1}{l})^d$-approximation algorithm (PTAS) with running time $O(l^d (l\sqrt{d})^d (2n)^{d(l\sqrt{d})^d+1})$ for the CUDC problem, where d is a positive integer. Gonzalez [40] proposed a $2(1+\frac{1}{l})^{d-1}$-approximation algorithm with running time $O(l^{d-1} d \lceil 2\sqrt{d} \rceil \lceil l\sqrt{d} \rceil^{d-1} n^{d\lceil 2\sqrt{d} \rceil^{d-1}+1})$. However, the running time of these algorithms is impractical for large input. Gonzalez [40] also presented an 8-approximation algorithm for the CUDC problem. The running time of this algorithm is $O(n \log |OPT|)$, where $|OPT| \leq n$ is the number of disks in an optimal solution. For the CUDC problem in L_1 and L_∞ metric, Gonzalez [40] proposed a 2-approximation algorithm in $O(n \log |OPT|)$

time. Using the concept of ϵ-net, an $O(1)$-approximation algorithm with running time $O(n^3 \log n)$ is presented in [8]. However, the exact value of the approximation factor is not attempted to be determined. Fu et al. [29] presented a 2.8334-approximation algorithm with running time $O(n(\log n \log\log n)^2))$ for the CUDC problem. Using a different approach of dividing the plane into vertical strips of height $\sqrt{3}$, Liu and Lu [63] developed a $\frac{25}{6}$-approximation algorithm with running time $O(n \log n)$ for the CUDC problem. Recently, Biniaz et al. [9] presented a 4-approximation algorithm for the CUDC problem, which runs in $O(n \log n)$ time. A slight variation of the CUDC problem is studied by Franceschetti et al. [30]. By constraining the centers of the disks to the vertices of a grid and using the shifting lemma [50], Franceschetti et al. [30] developed a $3(1 + \frac{1}{l})^2$-approximation algorithm, where $l \geq 1$. The running time of this algorithm is $O(Kn)$, where K is a function of l and the grid size. A listing of all the algorithms together with their approximation factors and running times for the CUDC problem in plane is given in Table 2.2.

Approximation factor	Running time	Reference
$(1 + \frac{1}{l})^2$	$O(l^4 n^{4l^2+1})$	Hochbaum & Maass, 1985
$2(1 + \frac{1}{l})$	$O(l^2 n^7)$	Gonzalez, 1991
8	$O(n \log \lvert OPT \rvert)$	Gonzalez, 1991
$O(1)$	$O(n^3 \log n)$	Brönnimann & Goodrich, 1995
2.8334	$O(n(\log n \log\log n)^2))$	Fu et al., 2007
$\frac{25}{6}$	$O(n \log n)$	Liu & Lu, 2014
4	$O(n \log n)$	Biniaz et al., 2015

Table 2.2: Approximation algorithms for the CUDC problem

A *sector* is a maximal region formed by the intersection of a set of disks i.e., all the points within the sector are covered by the same set of disks. Funke et al. [32] proposed the *greedy sector cover* algorithm for the *rectangular region cover* (RRC) problem. The approximation factor of their algorithm is $O(\log w)$, where w is the maximum number of sectors covered by a single disk. They proved that the greedy sector cover algorithm has an approximation factor no better than $\Omega(\log w)$. In the same paper, they proposed the grid placement algorithm (based on the algorithm proposed by Bose et al. [10]) and proved that their algorithm produces an 18π-approximation result. Though the algorithm is not guaranteeing full coverage of the region of interest, the area that remains uncovered can be bounded by a chosen number of grids. In the same paper, they have also considered the RRC problem in a different setup. We denote this setup as *reduced radius* setup. Here, we assume that the region of interest $\mathcal{R}$ is also covered by the disks in $\mathcal{D}$ after reducing their radius to $(1-\gamma)$. γ is said to be the *reduce radius parameter*. Reduce radius setup has many applications in wireless sensor networks, where coverage remains stable under small perturbations of sensing ranges/positions. In this setup an algorithm $\mathcal{A}$ is said to be a β-approximation if $\frac{|\mathcal{A}_{out}|}{|opt|} \leq \beta$, where $\mathcal{A}_{out}$ is the output of algorithm $\mathcal{A}$ and *opt* is the optimum set of disks with reduced radius such that the union of the disks in *opt* with reduced radius covers the region of interest. In reduce radius setup, Funke et al. [32] proposed a 4-approximation algorithm for the RRC problem. In this thesis, we have proposed a PTAS for the RRC problem in reduce radius setup.

Gandhi et al. [37] studied another variation of the covering problem called partial covering. In partial covering, unlike in the standard covering problems, it is desired to cover only a certain number of elements rather than covering all elements. For example, in k-set cover, we have to find the minimum number of sets to cover at least k elements. For set-cover, where each set has cardinality at most 3, they have given a $\frac{4}{3}$-approximation algorithm for the partial coverage. In the geometric covering problem for the partial coverage case, we are given n points in a d-dimensional space, we have to find the smallest number of identical disks of diameter Λ that cover at least k points. Using a shifting strategy [50], a PTAS (i.e. $(1+\epsilon)$-factor approximation result) in $O(\frac{1}{\epsilon^2}k^2n^{\frac{4}{\epsilon^2}+2})$ time is available in the literature [37]. In [39], the dual of this problem called the most points covering problem has been discussed. In this problem, there are n points in $I\!R^2$, and we have to cover a maximum number of points using m disks with radius r ($m > 0$ and $r > 0$). Both the partial covering problem and the most points covering problem are NP-hard [39]. Given n points in $I\!R^2$, a parameter $0 < \epsilon < 1$, and an integer $0 \leq k \leq n$, which is the number of points to be covered, there is an $(1+\epsilon)$-approximation algorithm with running time $O(\frac{1}{\epsilon}kn^{\frac{4}{\epsilon}+1})$ for the partial covering problem [39], which improves the running time of $O(\frac{1}{\epsilon^2}k^2n^{\frac{4}{\epsilon^2}+2})$ [37]. For the most points covering problem, using the algorithm for the partial covering problem, a $(1-\frac{2\epsilon}{1+\epsilon})$-approximation algorithm in $O((1+\epsilon)mn + \frac{1}{\epsilon}n^{\frac{4\sqrt{2}}{\epsilon}+2})$ time is available [39].

The class cover problem is defined as follows: let $\mathcal{B}$ be the set of blue points in class one and $\mathcal{R}$ be the set of red points in class two in a d-dimensional space. The goal is to cover the blue points with a minimum cardinality set of blue-centered balls of equal radius such that no red points lie in these balls. Therefore, 0-radius disks centered at all blue points gives a trivial solution. Cannon and Cowen [13] showed that this problem is NP-complete and provided a $(\ln n + 1)$-approximation algorithm which runs in $O(n^3 + dn^2)$ time, where $|\mathcal{R} \cup \mathcal{B}| = n$ and d is the dimension.

Kartz and Morgestern [58] studied a related geometric cover problem. In their problem, a set of m points $\mathcal{Q}$ is contained in a simple polygon P with n vertices. The objective is to compute a minimum cover of $\mathcal{Q}$ by disks contained in P. They gave a $O(nm^2)$ time algorithm for this problem. Later, Kaplan et al. [57] presented an almost linear time algorithm for this problem. The running time of their algorithm is $O(n + m(\log n + \log^2 m))$.

Sun and Lai [75] introduced another variation of the disk cover problem as follows: let $\Delta = \{d_0, d_1, \ldots, d_n\}$ be a set of disks of radius r with all their centers located inside d_0. Given Δ, the minimum disk cover problem seeks to identify a minimum subset of Δ, say Δ', such that the union of the disks in Δ' is equal to the union of the disks in Δ. They proposed an algorithm to solve the problem optimally in $O(n^{4/3})$ time [75]. Later, Sun et al. [77] proposed an optimal algorithm in $O(n \log n)$ time for the same problem using a divide-and-conquer strategy.

Given a set of r red points, a set of n blue points and a set of m objects, Chan and Hu [17] considered the problem of computing the smallest number of objects covering all blue points, while minimizing the number of red points covered by these objects. They proved that the problem is NP-hard even when the objects are unit squares. They proposed an $(1 + \epsilon)$-approximation algorithm (PTAS) for the problem, where the objects are unit squares $(0 < \epsilon \leq 1)$. The problem is a *discrete unit square cover* (DUSC) problem if the red point set is empty. In the DUSC problem, a set $\mathcal{P}$ of n points and a set $\mathcal{S}$ of m axis-parallel unit squares (side length is unit) are given, and the objective is to choose a minimum cardinality subset $\mathcal{S}^* \subseteq \mathcal{S}$ such that the union of the squares in $\mathcal{S}^*$ covers all the points in $\mathcal{P}$. The fastest algorithm is obtained by putting $\epsilon = 1$ in their PTAS to get 2-approximation algorithm in $O(m^{324}n)$ time, which is not practical [17]. As mentioned earlier, Mustafa and Ray proposed an $(1 + \epsilon)$-approximation algorithm

$(0 < \epsilon \leq 2)$ for the DUDC problem [66]. This PTAS is applicable for the DUSC problem also. Here, the fastest algorithm is obtained by putting $\epsilon = 2$ in their PTAS to get a 3-approximation algorithm in $O(m^{65}n)$ time, which is also not practical. Erlebach and van Leeuwen also have given an $(1+\epsilon)$-approximation algorithm (PTAS) for the DUSC problem, where $0 < \epsilon \leq 1$ [78]. Their algorithm is based on a very complicated dynamic programming paradigm. For $\epsilon = 1$, their PTAS provides the fastest algorithm, which is a 2-approximation algorithm with running time $O(m^8n^2)$ [62]. In this thesis we have proposed a $(2 + \frac{4}{k-2})$-approximation algorithm, which runs in $O(km^kn)$ time and a 2-approximation algorithm, which runs in $O(m^4n + n\log n)$ time for the DUSC problem, where $k(> 2)$ is an integer parameter that defines a trade-off between the running time and the approximation factor of the algorithm. Ito et al. [53] employed a similar dynamic programming approach based on the plane sweep technique of [78] to solve the *unique unit square coverage* problem. In the *unique unit square coverage* problem, given a set $\mathcal{P}$ of points and a set $\mathcal{S}$ of axis-aligned unit squares in $I\!R^2$, the objective is to find a subset $\mathcal{S}^* \subseteq \mathcal{S}$ of squares that maximizes the number of points of $\mathcal{P}$ contained in exactly one square in $\mathcal{S}^*$. For the *unique unit square coverage* problem, Ito et. al [53] proposed a $\frac{1}{1+\epsilon}$-approximation algorithm for any fixed constant $\epsilon > 0$, which improved the previous $\frac{1}{2}$-approximation algorithm by van Leeuwen [62].

Given a set $\mathcal{P}$ of n points in $\mathbb{R}^2$, Mahapatra et al. [65] considered the problem of computing two isothetic unit squares such that they cover the maximum number of points. They gave an algorithm that runs in $O(n^2)$ time using $O(n^2)$ space. They also considered the problem of computing k disjoint unit squares which maximizes the sum of points covered by them. For this problem, they gave an algorithm that runs in $O(k^2n^5)$ time using $O(kn^4)$ space. They also gave an $O(n\log n)$ time and $O(n)$ space algorithm which computes $O(n)$ isothetic unit squares covering the maximum number of points with each having one side aligned with a given point. Saha et al. [74] considered the problem of finding two parallel rectangles with arbitrary orientation to cover a given set of n points in $\mathbb{R}^2$ such that the area of the larger rectangle is minimized. For this problem, they proposed an algorithm that runs in $O(n^3)$ time using $O(n^2)$ space.

Given a set $\mathcal{P}$ of n points in $I\!R^2$, the classical *k-center* problem is to cover $\mathcal{P}$ with k congruent disks of radius as small as possible. The above problem is called the *dis-*

crete k-center problem when the centers of the k disks are restricted to be chosen from a set of points, otherwise it is called the *continuous k-center* problem or simply the *k-center* problem. Both versions of the k-center problem are NP-complete for $k \geq 2$ if k is part of the input [5]. The various constrained versions of the *k-center* problem have been studied extensively in the literature. Hurtado et al. [47] considered the Euclidean 1-center problem where the center is constrained to satisfy m linear constraints, and proposed an $O(n + m)$ time algorithm for it. Bose and Toussaint [12] provided an $O((n + m)\log(n + m))$ time algorithm for the 1-center problem, where the disk is centered on the boundary of a convex polygon with m vertices, and the objective is to cover n demand points that may lie inside or outside of the polygon. Brass et al. [11] studied a similar version of the *k-center* problem, where the centers are constrained to lie on a given straight line. It uses parametric search, and runs in $O(n \log^2 n)$ time. Karmakar et al. [59] proposed three algorithms for this problem with time complexities $O(nk \log n)$, $O(nk + k^2 \log^3 n)$ and $O(n \log n + k \log^4 n)$, respectively. Using parametric search, Kim and Shin [61] solved the 2-center problem for a given polygon in $O(n \log^2 n)$ time, where the two centers are restricted to be at some vertices of the polygon. Halperin et al. [51] also considered a version of 2-center problem where the two centers are restricted to lie outside the boundaries of disjoint simple polygons with a total of m edges. For this version of the 2-center problem, Halperin et al. [51] gave an algorithm with expected time $O(m \log^2(mn) + mn \log^2 n \log(mn))$ and a $(1+\epsilon)$-approximation algorithm in time $O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon})(m \log^2 m + n \log^2 n))$ or in randomized expected time $O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon})((m+n\log n)\log(mn)))$, where $\epsilon > 0$ and n is the number of points to be covered. Suzuki and Drezner [73] investigated the p-center problem for demand originating in an area and proposed heuristic procedures for the problem. Das et al. [24] provided a $(1 + \epsilon)$-approximation algorithm for the k-center problem on a convex polygon, where the centers are restricted to lie on a specified edge of the polygon. If the centers are restricted to be on the boundary of a convex polygon, Das et al. [24] presented an $O(n^2)$ time algorithm respectively for $k = 1, 2$. In the same paper, they presented a heuristic algorithm for the same problem for $k \geq 3$. Later, Roy et al. [71] improved the time complexities of the same problem for $k = 1, 2$ to $O(n)$. Du and Xu [25] studied the *k-center* problem for a convex polygon where the centers are restricted to lie on the boundary of the polygon only, and presented a 1.8841-approximation algorithm, which runs in $O(nk)$ time, where n is the number of vertices of the polygon. In this thesis we have given a $(1 + \frac{7}{k} + \frac{7\epsilon}{k} + \epsilon)$-approximation algorithm with running time $O(n(n + k)(|\log r_{opt}| + \log\lceil\frac{1}{\epsilon}\rceil))$ for the same problem, where r_{opt} is the radius of disks in the optimal solution[1], $k \geq 7$, and $\epsilon > 0$ is any given constant.

Hochbaum and Shmoys [44], and Gonzalez [42] provided 2-approximation algorithms for the *k-center* problem under general metrics. The running time of their algorithms are $O(n^2 \log n)$ and $O(nk)$ respectively. This is the best possible approximation bound as it is NP-hard to approximate beyond a factor of 2 for the *k-center* problem under general metrics [46]. However, Feder and Greene [36] gave a 2-factor approximation algorithm in $O(n \log k)$ time for the *Euclidean k-center* problem and showed that on Euclidean metrics, this problem cannot be approximated within a factor of $\sqrt{3} \approx 1.73$ unless $P = NP$.

A generalization of the *k-center* problem, namely the *k-supplier* problem is available in the literature. In the *k-supplier* problem, the set of given points is partitioned into two subsets $\mathcal{Q}$ (*facilities*) and $\mathcal{P}$ (*clients*), and the objective is to choose k facilities such that the maximum distance of any client to its nearest chosen facility is minimum. In a general metric, Hochbaum and Shmoys [45] gave a 3-approximation algorithm running in $O((n^2 + mn)\log(mn))$ time, where m is the number of facilities and n is the number of clients. They also proved that a $(3 - \epsilon)$-approximation algorithm in polynomial time is not possible unless $P = NP$. However, in the Euclidean metric, Feder and Greene [36] gave a 3-approximation algorithm for the *Euclidean k-supplier* problem with running time $O((n + m) \log k)$. They also showed that it is NP-hard to approximate the *Euclidean k-supplier* problem less than a factor of $\sqrt{7} \approx 2.64$. Furthermore, for fixed k, Hwang et al. [49] presented a $m^{O(\sqrt{k})}$-time algorithm for the *Euclidean k-supplier* problem. Later, Agarwal and Procopiuc [4] gave $m^{O(k^{1-1/d})}$-time algorithm for d-dimensional points. Recently, Nagarajan et al. [69] gave a 2.74-approximation algorithm for the *Euclidean k-supplier* problem in any constant dimension with running time $O(mn \log(mn))$. In this thesis, we have proposed a FPT 2-approximation algorithm for the *Euclidean k-supplier* problem in plane. The running time of this FPT 2-approximation algorithm is

[1] we have taken $|\log r_{opt}|$ in the time complexity since r_{opt} may be less than 1

$O(6^k(n+m)\log(mn))$. We generalize our FPT 2-approximation algorithm to develop a FPT $(1+\epsilon)$-approximation algorithm for the k-supplier problem in $I\!R^d$, where d is a positive integer and $\epsilon > 0$ is an arbitrary number. The running time of the proposed $(1+\epsilon)$-approximation algorithm is $O(\epsilon^{-dk}(m+n)\log(mn))$. We also propose a heuristic algorithm for the Euclidean k-supplier problem in $I\!R^2$ and experimentally show that the proposed heuristic performs very well for randomly generated instances.

Recently, Dumitrescu and Jiang [26] studied the following variation of the *constrained k-center* problem: given a set $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ of n black points and a set $\mathcal{Q} = \{q_1, q_2, \ldots, q_k\}$ of k red points in $I\!R^2$, the goal is to find a set $\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$ of k disks such that (i) the disk D_j must contain the red point $q_j \in \mathcal{Q}$ for $1 \leq j \leq k$, (ii) all points in $\mathcal{P}$ are covered by the union of the disks in $\mathcal{D}$ and (iii) the maximum radius of the disks in $\mathcal{D}$ is minimized. Dumitrescu and Jiang [26] showed that their *constrained k-center* problem is NP-hard and can not be approximated within a 1.8279 factor. They proposed FPT-algorithms with 1.87, 1.71, and 1.61-approximation results in $O(3^k kn)$, $O(4^k kn)$, and $O(5^k kn)$ time, respectively. Based on the generalization of the idea used for developing the above FPT-algorithms, they proposed an $(1+\epsilon)$-approximation algorithm in $O(\epsilon^{-2k}n)$ time, where $\epsilon > 0$.

There are several works slightly related to the domain of geometric covering, where the objective is to cover certain geometric objects with a set of other geometric objects which are pairwise interior disjoint [1, 2, 3]. In this setting one of the recently studied interesting problems is that of constructing a *cover contact graph* (CCG). In the CCG problem the objective is to cover certain geometric objects called *seeds* (e.g. points) by a set of other geometric objects called *cover* (e.g. a set of disks, set of triangles, etc.). The interiors of *seeds* and *cover* elements are pairwise disjoint, respectively but they can touch. The contact graph of *cover* is denoted as *cover contact graph* (CCG). Atienza et al. [3] gave algorithms to decide whether a given set of points can be covered with disks or triangles such that the resulting *cover contact graph* is a 1- or 2- connected graph. Recently, Hossain et al. [52] gave an algorithm to construct a 4-connected *triangle cover contact graph* (TCCG) for a set of point seeds, where the *cover* is a set of triangles. There is enormous work in the general area of geometric covering. Let $\mathbb{U}$ be any universal set and let $\mathbb{M} \subset \mathbb{U}$ be any subset of it. Then, given a collection S of subsets of $\mathbb{U}$, the set cover problem is to find a minimum size subcollection $C \subseteq S$ that covers $\mathbb{M}$. Clarkson and Varadarajan [60] considered various geometric special cases of this problem, where $\mathbb{U} = \mathbb{R}^d$, and proved that several polynomial-time approximation algorithms exist for these geometric set cover problems. Somemore work on geometric set cover and related references can be found in [55, 28, 54, 56, 68].

Chapter 3

Discrete Unit Disk Cover Problem

In this chapter we consider the following two problems:

1. *Discrete unit disk cover* (DUDC) problem: Given a set $\mathcal{P}$ of n points and a set $\mathcal{D}$ of m unit disks in $I\!R^2$, the objective is (i) to check whether the union of all the disks in $\mathcal{D}$ covers all the points in $\mathcal{P}$, and (ii) if yes, then select a minimum cardinality subset $\mathcal{D}^* \subseteq \mathcal{D}$ such that each point in $\mathcal{P}$ is covered by at least one disk in $\mathcal{D}^*$.

2. *Rectangular region cover* (RRC) problem: Given a rectangular region $\mathcal{R}$ and a set $\mathcal{D}$ of m unit disks in $I\!R^2$, the objective is (i) to check whether the union of all the disks in $\mathcal{D}$ covers $\mathcal{R}$, and (ii) if so, then select a minimum cardinality subset $\mathcal{D}^{**} \subseteq \mathcal{D}$ such that each point of a given rectangular region $\mathcal{R}$ is covered by the union of the disks in $\mathcal{D}^{**}$.

The DUDC problem is a well-studied problem in the domain of geometric covering. In the literature, there are many approximation algorithms based on various techniques for the DUDC problem. The PTAS of Mustafa and Ray [66], which uses the concept of ϵ-net and local search technique, is useful only for $1 < \epsilon \leq 2$. For each value of ϵ, their PTAS takes a huge amount of time. The other approximation algorithms [6, 15, 19, 20, 70, 23, 33] are all mostly based on the technique of dividing the plane into smaller chucks such as triangles, squares and horizontal strips of fixed height, then finding the cover for points lying in these chunks. In this chapter, we improve upon the previous results for the DUDC problem by computing a better cover for points lying in one such kind of chunks (LSDUDC). Unlike the PTAS of Mustafa and Ray [66], our algorithm runs in a reasonable running time for the approximation range $9 + \epsilon$, where $0 < \epsilon \leq 6$.

The objective of this chapter is aimed at developing constant factor approximation algorithms with reasonable running time for the DUDC and RRC problems. In order to solve the DUDC problem, we consider two subproblems of the DUDC problem, namely (i) the *line separable discrete unit disk cover* (LSDUDC) problem, and (ii) the *within strip discrete unit disk cover* (WSDUDC) problem. In this chapter we propose an $(1 + \mu)$-approximation algorithm (PTAS) for the LSDUDC problem, where $0 < \mu \leq 1$. The running time of the $(1+\mu)$-approximation algorithm is $O(m^{3(1+\frac{1}{\mu})} n \log n)$. Based on the PTAS for the LSDUDC problem, and a 3-approximation algorithm for the WSDUDC problem proposed by Fraser and López [33], we propose a $(9+\epsilon)$-approximation algorithm for the DUDC problem, where $0 < \epsilon \leq 6$. The running time of the proposed algorithm for the DUDC problem is $O(m^{3(1+\frac{6}{\epsilon})} n \log n)$. Using this $(9+\epsilon)$-approximation algorithm for the DUDC problem, we propose a $(9 + \epsilon)$-approximation algorithm for the RRC problem. The running time of the algorithm is $O(m^{5+\frac{18}{\epsilon}} \log m)$. We also consider the RRC problem in reduce radius setup. For the RRC problem in reduce radius setup, we develop an $(1 + 1/l)^2$-approximation algorithm (PTAS) using the shifting strategy developed by Hochbaum and Maass [50], where l is a positive integer. The running time of the algorithm is $O(ql^2 2^{\lceil \frac{4l^2}{\nu^2} + \frac{8l+4}{\nu} \rceil})$, where q is the minimum number of squares of size $2l \times 2l$, whose union covers the region $\mathcal{R}$, and $\nu = \sqrt{2}\gamma$, where γ is called the *radius ruduction parameter*.

In Section 3.1, we sketch the outline of previous algorithms for the DUDC problem. In Section 3.2, we propose a PTAS for the LSDUDC problem. We present an approximation algorithm for the DUDC problem using the proposed PTAS for the LSDUDC problem in Subsection 3.2.1. Approximation algorithms for RRC problems are presented in Section 3.3. Finally, we conclude the chapter in Section 3.4.

3.1 Previous Approximation Algorithms for DUDC Problem

Das et al. [23] first described a procedure for testing the feasibility of the DUDC problem using the nearest point Voronoi diagram and a planar point location algorithm. Given a set $\mathcal{P}$ of n points and a set $\mathcal{D}$ of m unit disks, the feasibility of the DUDC problem can be checked in $O(m \log m + n \log m)$ time using the procedure described below.

Let $\mathcal{Q}$ be the set of centers of unit disks in $\mathcal{D}$ and $dist(a, b)$ denote the Euclidean distance between two points a and b.

1) Compute the nearest point Voronoi diagram $VOR(\mathcal{Q})$ of the points in $\mathcal{Q}$ [7].

2) Invoke a planar point location algorithm in the planar subdivision $VOR(\mathcal{Q})$ for each point $p \in \mathcal{P}$, and find its nearest point $q_p \in \mathcal{Q}$.

3) If $dist(p, q_p) \leq 1$ for all $p \in \mathcal{P}$, then all the points in $\mathcal{P}$ are covered by the union of unit disks in $\mathcal{D}$, hence the given instance of DUDC problem has a feasible solution, otherwise the instance has no feasible solution.

Now, we define three subproblems which are very useful to describe the DUDC problem, namely (i) *line-separable* DUDC (LSDUDC) problem, (ii) *within-strip* DUDC (WSDUDC) problem and (iii) *outside-strip* DUDC (OSDUDC) problem as follows:

(i) In the LSDUDC problem, the plane being divided into two half planes ℓ^+ and ℓ^- defined by a line ℓ, all the points in $\mathcal{P}$ are in ℓ^- and the centers of the disks in $\mathcal{D}$ are in $\ell^+ \cup \ell^-$ such that each point in $\mathcal{P}$ is covered by at least one disk centered in ℓ^+. The objective is to select a minimum cardinality subset $\mathcal{D}^* \subseteq \mathcal{D}$ such that each point in $\mathcal{P}$ is covered by at least one disk in $\mathcal{D}^*$.

(ii) In the WSDUDC problem, all points in $\mathcal{P}$ and centers of all disks in $\mathcal{D}$ are inside a horizontal strip of height $\frac{1}{\sqrt{2}}$. The objective is to select a minimum cardinality subset $\mathcal{D}^* \subseteq \mathcal{D}$ such that each point in $\mathcal{P}$ is covered by at least one disk in $\mathcal{D}^*$.

(iii) In the OSDUDC problem, a set $\mathcal{P}$ of points are lying inside horizontal strips $\mathcal{H}$ of height $\frac{1}{\sqrt{2}}$ and all points in $\mathcal{P}$ are covered by the disks in $\mathcal{D}$ such that the centers of the disks in $\mathcal{D}$ are outside the horizontal strips $\mathcal{H}$, the objective is to select a minimum cardinality subset $\mathcal{D}^* \subseteq \mathcal{D}$ such that each point in $\mathcal{P}$ is covered by at least one disk in $\mathcal{D}^*$.

Let $\mathcal{R}$ be the minimum-sized axis-aligned rectangle containing all points in $\mathcal{P}$ and centers of all disks in $\mathcal{D}$. The rectangle $\mathcal{R}$ is divided into horizontal strips bounded by the line segments $\ell_0, \ell_1, \ldots, \ell_t$ indexed from top to bottom such that $dist(\ell_i, \ell_{i+1}) = \frac{1}{\sqrt{2}}$ for $i = 0, 1, \ldots, t-1$ where the top and bottom boundaries of the rectangle $\mathcal{R}$ are denoted by the line segments ℓ_0 and ℓ_t respectively and $dist(a, b)$ is used to denote the Euclidean distance between two horizontal line segments a and b. Let $[\ell_i, \ell_{i+1}]$ denote the horizontal strip defined by the lines ℓ_i and ℓ_{i+1}.

Based on the t horizontal strips, Das et al.[23] partitioned the point set $\mathcal{P}$ into 7 disjoint sets and solved them independently, which involves both the OSDUDC problem and the WSDUDC problem.

Lemma 3.1.1. *[23] The approximation factor of the OSDUDC algorithm is* $6\times$ *(approximation factor of the LSDUDC algorithm), and the running time of the OSDUDC algorithm is* $O(n \log n + mn)$*.*

Lemma 3.1.2. *[23] The approximation factor of the WSDUDC algorithm is* 6*, and the running time of the algorithm is* $O(n \log n + mn)$*.*

Theorem 3.1.3. *[23] The approximation factor of the DUDC algorithm is the sum of the approximation factor of the OSDUDC algorithm and the approximation factor of the WSDUDC (defined on strip of height* $\frac{1}{\sqrt{2}}$*) algorithm, and the running time of the DUDC algorithm is* $O(\max($*running time of LSDUDC algorithm, running time of WSDUDC algorithm*$))$*.*

Corollary 3.1.4. *The approximation factor of the algorithm for the DUDC problem is* $6 \times 2 + 6 = 18$ *and the running time of the algorithm is* $O(m \log m + n \log n + mn)$*.*

The approximation factor of the DUDC algorithm can be improved using the following improved results for the WSDUDC problem.

Lemma 3.1.5. *[33] The WSDUDC problem admits a* $3\lceil \frac{1}{\sqrt{1-\delta^2}} \rceil$*-approximation algorithm in* $O(m^4 n + n \log n)$ *time for a strip of height* $\delta < 1$*.*

Lemma 3.1.6. *[33] The WSDUDC problem admits a 4-approximation algorithm in $O(m^4 n + n \log n)$ time for a strip of height $\delta \leq 2\frac{\sqrt{2}}{3}$.*

Lemma 3.1.7. *[33] The WSDUDC problem admits a 3-approximation algorithm in $O(m^6 n + n \log n)$ time for a strip of height $\delta \leq \frac{4}{5}$.*

Theorem 3.1.8. *[33] The DUDC problem admits a 15-approximation algorithm in $O(m^6 n + n \log n)$ time.*

Proof. Follows from Theorem 3.1.3, Lemmata 3.1.1 and 3.1.7, and since $\frac{1}{\sqrt{2}} < \frac{4}{5}$. □

3.2 PTAS for the LSDUDC Problem

In this section we first define some terminology as follows:
Let ℓ be a horizontal line. We use ℓ^+ and ℓ^- to denote the half-planes above and below ℓ respectively. The definition of the LSDUDC problem is as follows:

> A set $\mathcal{P}$ of points and a set $\mathcal{D}$ of unit disks exist such that (i) each point in $\mathcal{P}$ is in ℓ^-, (ii) the center of each disk in $\mathcal{D}$ is in $\ell^+ \cup \ell^-$, and (iii) the union of the disks centered in ℓ^+ covers all points in $\mathcal{P}$. The objective is to find a minimum cardinality set $\mathcal{D}^* \subseteq \mathcal{D}$ such that the union of the disks in $\mathcal{D}^*$ covers $\mathcal{P}$ i.e., $\mathcal{P} \subseteq \bigcup_{d \in \mathcal{D}^*} d$.

We use $\mathcal{U}$ and $\mathcal{L}$ to denote the set of disks in $\mathcal{D}$ with centers in ℓ^+ and ℓ^- respectively. For a disk $d \in \mathcal{D}$, its boundary arc and center are denoted by $\theta(d)$ and $\alpha(d)$ respectively. A disk $d \in \mathcal{U}$ is said to be a *lower boundary disk* if there does not exist $X = \mathcal{U} \setminus \{d\}$ such that $d \cap \ell^- \subset (\bigcup_{d' \in X} d') \cap \ell^-$. For a lower boundary disk $d \in \mathcal{U}$, we use the term *lower region* to denote the region $d \cap \ell^-$ and *lower arc* to denote the arc $\theta(d) \cap \ell^-$ (see Figure 3.1). We use $\mathcal{D}_\ell = \{d_1, d_2, \ldots, d_s\} \subseteq \mathcal{U}$ to denote the set of all lower boundary disks. We use B_{region} to denote the union of lower regions covered by the disks in $\mathcal{D}_\ell$ i.e., $B_{region} = (\bigcup_{d' \in \mathcal{D}_\ell} d') \cap \ell^-$.

Let L be an arbitrary vertical line. We use L^- (resp. L^+) to denote the region in the left (resp. right) side of the vertical line L. Let $\mathcal{P}_{L^-}$ (resp. $\mathcal{P}_{L^+}$) be the set of points in $\mathcal{P}$ to the left (resp. right) of L i.e., $\mathcal{P}_{L^-} = \mathcal{P} \cap L^-$ and $\mathcal{P}_{L^+} = \mathcal{P} \cap L^+$. Let $\mathcal{D}^{L-} (\subseteq \mathcal{D})$ and $\mathcal{D}^{L+} (\subseteq \mathcal{D})$ be the optimum cover of the points in $\mathcal{P}_{L^-}$ and $\mathcal{P}_{L^+}$ respectively.

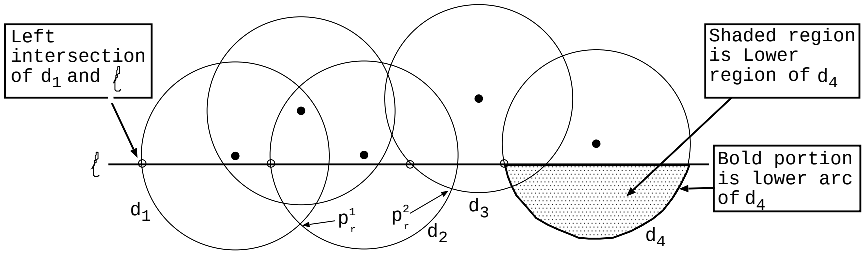


Figure 3.1: Lower region, left intersection and lower boundary disks

Claim 3.2.1. For two disks $d', d'' \in \mathcal{L}$, if $d', d'' \in \mathcal{D}^{L-}$ and $d', d'' \in \mathcal{D}^{L+}$, and d' and d'' intersect with each other, then both d' and d'' intersect L and both the intersections between $\theta(d')$ and $\theta(d'')$ lie in B_{region}.

Proof. Both the disks d' and d'' intersect L because $d', d'' \in \mathcal{D}^{L-}$ and $d', d'' \in \mathcal{D}^{L+}$.

Since $d', d'' \in \mathcal{D}^{L-}$ and $d', d'' \in \mathcal{D}^{L+}$, there exist points $p'_0, p''_0 \in \mathcal{P}_{L^-}$ and $p'_1, p''_1 \in \mathcal{P}_{L^+}$ such that $p'_0, p'_1 \in d'$ and $p''_0, p''_1 \in d''$ but $p'_0, p'_1 \notin d''$ and $p''_0, p''_1 \notin d'$ (see Figure 3.2). Now, if at least one intersection of d' and d'' lies below B_{region}, then either (i) one of p'_0 or p'_1 lies outside B_{region} or (ii) one of p''_0 or p''_1 lies outside B_{region}, which leads to a contradiction because there must be a point either in L^- or in L^+ which is covered by one of d' or d'' but not by the other or vice versa for both d', d'' to be in $\mathcal{D}^{L-}$ and $\mathcal{D}^{L+}$ simultaneously and at least one of the intersection points between $\theta(d')$ and $\theta(d'')$ lie below B_{region}, but each point in $\mathcal{P}$ is covered by at least one disk centered above ℓ.

Now, if $\theta(d')$ and $\theta(d'')$ intersect above ℓ, then either (i) $\mathcal{P}_{L^-} \cap d' \subset \mathcal{P}_{L^-} \cap d''$ or $\mathcal{P}_{L^-} \cap d'' \subset \mathcal{P}_{L^-} \cap d'$ or (ii) $\mathcal{P}_{L^+} \cap d' \subset \mathcal{P}_{L^+} \cap d''$ or $\mathcal{P}_{L^+} \cap d'' \subset \mathcal{P}_{L^+} \cap d'$ (Note: the centers of the disks d' and d'' lie below the line ℓ as $d', d'' \in \mathcal{L}$). Therefore, both d' and d'' cannot appear in the solutions $\mathcal{D}^{L-}$ and $\mathcal{D}^{L+}$, which leads to a contradiction. Thus, $\theta(d')$ and $\theta(d'')$ intersect in B_{region}. □

Definition 3.2.2. A pair $(d', d'')(\in \mathcal{L} \times \mathcal{L})$ of disks is said to be a weak (resp. strong) cover pair if $\theta(d')$ and $\theta(d'')$ intersect once (resp. twice) in B_{region}.

Definition 3.2.3. A pair $(d', d'')(\in \mathcal{L} \times \mathcal{L})$ of disks is said to be a non-intersecting cover pair if $\theta(d')$ and $\theta(d'')$ do not intersect with each other.

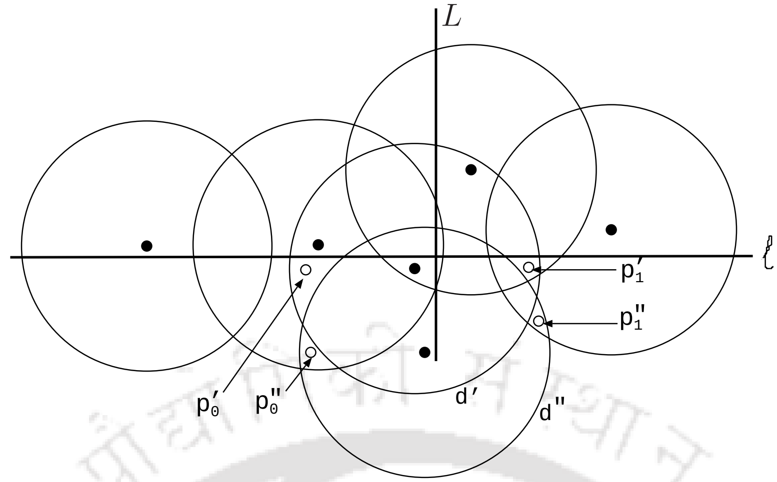


Figure 3.2: Proof of Claim 3.2.1

Lemma 3.2.4. *For a weak cover pair* $(d', d'')(\in \mathcal{L} \times \mathcal{L})$*;* $d', d'' \in \mathcal{D}^{L-}$ *and* $d', d'' \in \mathcal{D}^{L+}$ *cannot happen simultaneously.*

Proof. On contrary, assume $d', d'' \in \mathcal{D}^{L-}$ and $d', d'' \in \mathcal{D}^{L+}$. By the definition of weak cover pair, one of the intersections of $\theta(d')$ and $\theta(d'')$ lies in B_{region}, whereas the other intersection lies either above ℓ or below B_{region}. But, by Claim 3.2.1 the intersection point cannot lie below B_{region}. Therefore, either (i) $\mathcal{P}_{L^-} \cap d' \subset \mathcal{P}_{L^-} \cap d''$ or $\mathcal{P}_{L^-} \cap d'' \subset \mathcal{P}_{L^-} \cap d'$ or (ii) $\mathcal{P}_{L^+} \cap d' \subset \mathcal{P}_{L^+} \cap d''$ or $\mathcal{P}_{L^+} \cap d'' \subset \mathcal{P}_{L^+} \cap d'$ (see Figure 3.3). Thus, both the disks d', d'' cannot be in $\mathcal{D}^{L-}$ and $\mathcal{D}^{L+}$ simultaneously. □

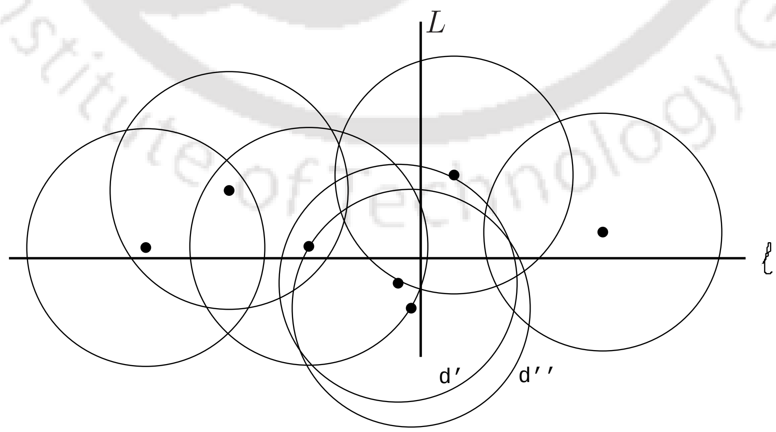


Figure 3.3: Proof of Lemma 3.2.4

Lemma 3.2.5. *For a strong cover pair $(d', d'')(\in \mathcal{L} \times \mathcal{L})$ with $\alpha(d')$ above $\alpha(d'')$, if the intersections of $\theta(d')$ and $\theta(d'')$ lie within lower boundary disks d_x and d_y, then either one intersection occurs between $\theta(d_x)$ and $\theta(d')$, or one intersection occurs between $\theta(d_y)$ and $\theta(d')$ above the horizontal line ℓ.*

Proof. Without loss of generality assume that $\alpha(d')$ is above the intersection point of d' and d'' inside d_x. Let a and b be the two intersection points of $\theta(d_x)$ and $\theta(d')$. Therefore, $\alpha(d')$ should lie above at least one point among a and b. Assume $\alpha(d')$ lies above a. By symmetry, $\overline{\alpha(d'), a}$ and $\overline{\alpha(d_x), b}$ are parallel (see Figure 3.4). Thus, b must be above $\alpha(d_x)$ i.e., b must be above ℓ. □

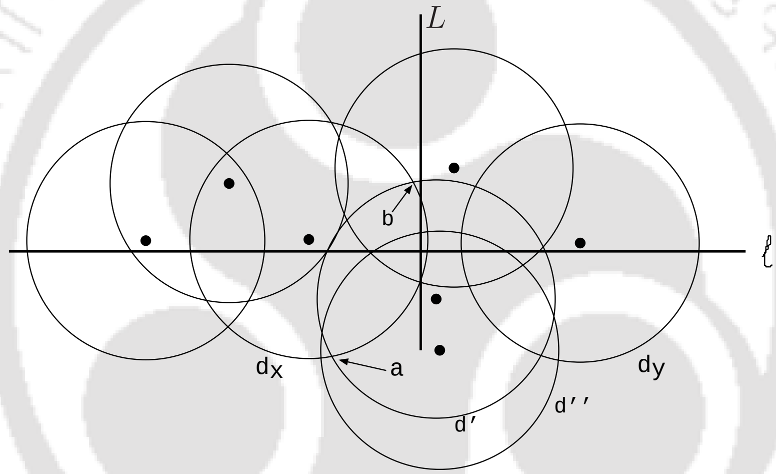


Figure 3.4: Proof of Lemma 3.2.5

Lemma 3.2.6. $|\mathcal{D}^{L-} \cap \mathcal{D}^{L+} \cap \mathcal{L}| \leq 2$.

Proof. On the contrary, assume that $d_x, d_y, d_z \in \mathcal{D}^{L-} \cap \mathcal{D}^{L+} \cap \mathcal{L}$. Since $d_x, d_y, d_z \in \mathcal{D}^{L-}$ as well as $d_x, d_y, d_z \in \mathcal{D}^{L+}$, all the disks d_x, d_y, d_z intersect the vertical line L in the B_{region}. Let $\Gamma = \{(d_x, d_y), (d_x, d_z), (d_y, d_z)\}$. From Lemma 3.2.4, no pair in Γ forms a weak cover pair because $d_x, d_y, d_z \in \mathcal{D}^{L-}$ as well as $d_x, d_y, d_z \in \mathcal{D}^{L+}$. Then, the following cases are possible:

(i) Every pair $(d_1, d_2) \in \Gamma = \{(d_x, d_y), (d_x, d_z), (d_y, d_z)\}$ forms a strong cover pair. Without loss of generality assume that $\alpha(d_x)$ is below $\alpha(d_y)$ and $\alpha(d_y)$ is below

$\alpha(d_z)$ (see Figure 3.5(a)). If a is the intersection between $\theta(d_x)$ and $\theta(d_y)$ inside the lower boundary disk d (say) and below the horizontal line ℓ, then from Lemma 3.2.5 one intersection between $\theta(d_y)$ and $\theta(d_z)$ lies inside of d (see Figure 3.5(a)). Therefore, $(d_x \cup d_y \cup d_z) \cap \mathcal{P}_{L^-} \subseteq (d_x \cup d) \cap \mathcal{P}_{L^-}$, which implies that $\mathcal{D}^{L-}$ is not optimum, leading to a contradiction, because the set $(\mathcal{D}^{L-} \setminus \{d_x, d_y, d_z\}) \cup \{d_x, d\}$, whose cardinality is smaller than $|\mathcal{D}^{L-}|$, covers $\mathcal{P}_{L^-}$.

(ii) If one pair (d_x, d_z)(say) in Γ forms a non-intersecting cover pair and the other two pairs (d_x, d_y) and (d_y, d_z) form strong cover pairs (see Figure 3.5(b)), let d be the lower boundary disk centered to the left of L (see Figure 3.5(b)). Then, one intersection between $\theta(d_x)$ and $\theta(d_y)$ lies inside of d as d_x does not intersect with d_z, and from Lemma 3.2.5 one intersection between $\theta(d_y)$ and $\theta(d_z)$ lies inside of d. Therefore, $(d_x \cup d_y \cup d_z) \cap \mathcal{P}_{L^-} \subseteq (d_z \cup d) \cap \mathcal{P}_{L^-}$, which implies that $\mathcal{D}^{L-}$ is not optimum, leading to a contradiction, because the set $(\mathcal{D}^{L-} \setminus \{d_x, d_y, d_z\}) \cup \{d_z, d\}$, whose cardinality is smaller than $|\mathcal{D}^{L-}|$, covers $\mathcal{P}_{L^-}$.

(iii) In this case, let's say only one pair $(d_y, d_z) \in \Gamma$ forms a strong cover pair and the other two pairs (d_x, d_y) and (d_x, d_z) form non-intersecting cover pairs (see Figure 3.5(c)). Let d be the lower boundary disk centered to the left of L (see Figure 3.5(c)). Then, $(d_x \cup d_y \cup d_z) \cap \mathcal{P}_{L^-} \subseteq (d_x \cup d) \cap \mathcal{P}_{L^-}$, which implies that $\mathcal{D}^{L-}$ is not optimum, leading to a contradiction.

(iv) The case when every pair in Γ forms a non-intersecting cover pair leads to contradiction by the same argument as in cases (ii) and (iii) (see Figure 3.5(d)).

The remaining cases are just the mirror cases (mirror case of (i) is depicted in Figure 3.6(a), mirror cases of (ii) are depicted in Figure 3.6(b), (c) and (d), mirror cases of (iii) are depicted in Figure 3.7(a), (b) and (c), and mirror case of (iv) is depicted in Figure 3.7(d)) that can be handled in the same way as above. Thus, the lemma follows.

□

Each disk in $\mathcal{D}_\ell$ intersects the horizontal line ℓ and B_{region} contains all points in $\mathcal{P}$. Without loss of generality assume that $d_1, d_2, \ldots, d_s$ is the sorted order from left to right based on their left intersection points with the line ℓ (see Figure 3.1). Since the centers

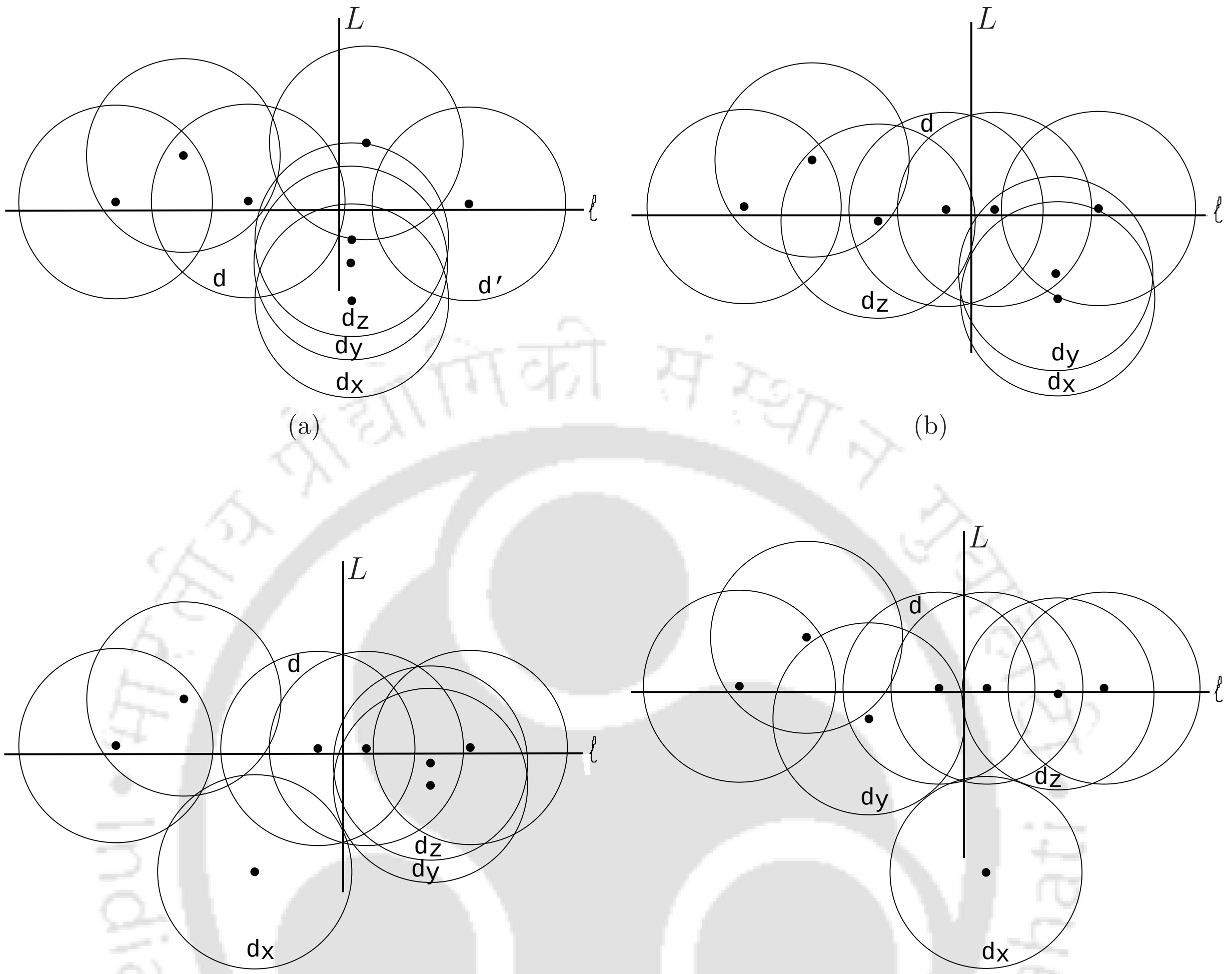


Figure 3.5: Proof of Lemma 3.2.6

of the disks in $\mathcal{D}_\ell$ are in ℓ^+, the number of intersection points (if any) between two disk arcs of $\mathcal{D}_\ell$ in ℓ^- is one. For each disk $d_i \in \mathcal{D}_\ell$ we define a point, namely p_r^i as follows:

p_r^i: If the disk d_i has intersection with d_{i+1} in ℓ^-, then p_r^i is the intersection point between $\theta(d_{i+1})$ and $\theta(d_i)$ in ℓ^-, otherwise p_r^i is the right intersection point between ℓ and $\theta(d_i)$.

Here d_0 and d_{s+1} are the two dummy disks having no intersection with d_1 and d_s respectively. For each $i = 1, 2, \ldots, s$, let $\mathcal{P}_i (\subseteq \mathcal{P})$ be the set of points lying between two vertical lines through p_r^{i-1} and p_r^i. Let e^i be the vertical line through the point p_r^i for

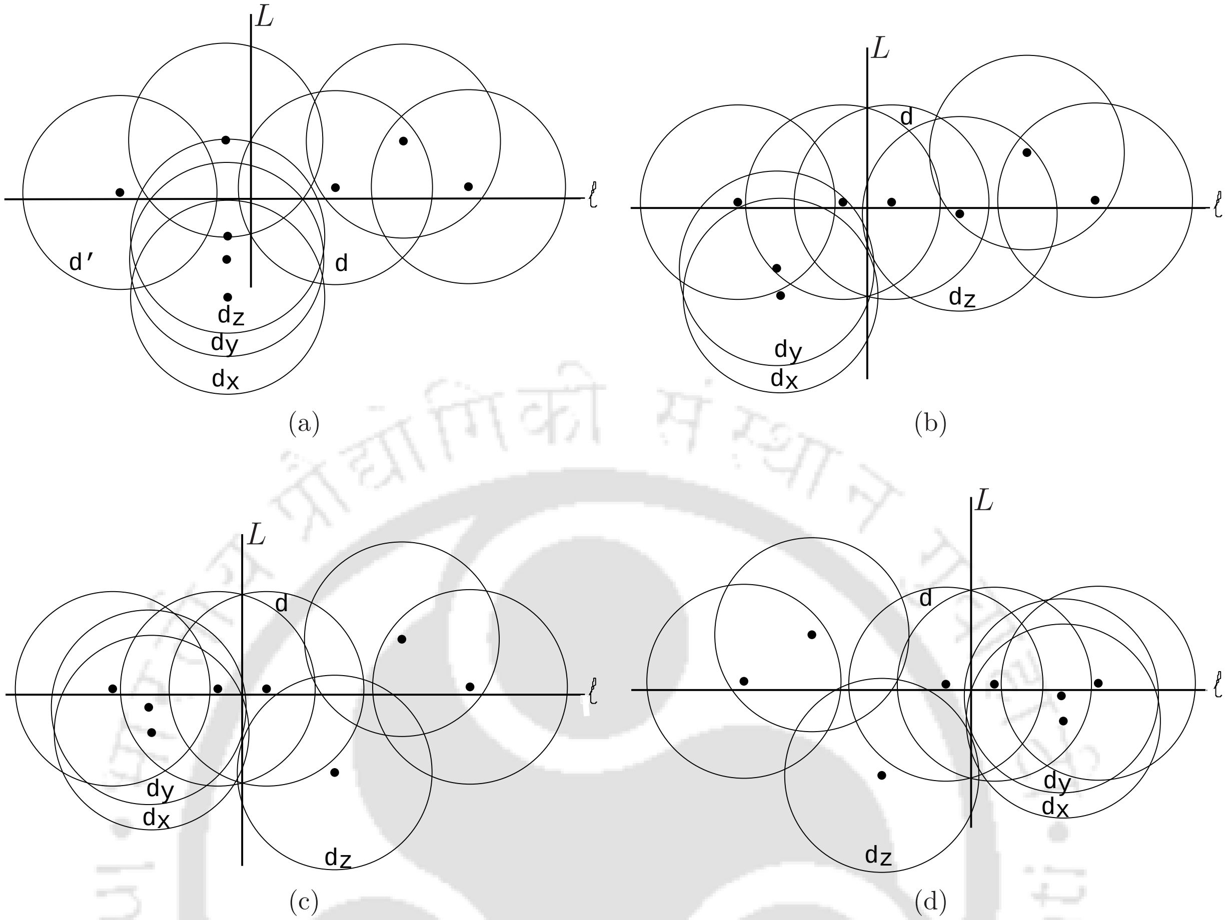


Figure 3.6: Proof of Lemma 3.2.6 (mirror cases)

$i = 1, 2, \ldots, s$. We use e^{i-} (resp. e^{i+}) to denote the region in the left (resp. right) side of the vertical line e^i. Let $\mathcal{P}_{e^{i-}}$ (resp. $\mathcal{P}_{e^{i+}}$) be the set of points in $\mathcal{P}$ to the left (resp. right) of e^i i.e., $\mathcal{P}_{e^{i-}} = \mathcal{P} \cap e^{i-}$ and $\mathcal{P}_{e^{i+}} = \mathcal{P} \cap e^{i+}$. Let $\mathcal{D}^{i-}(\subseteq \mathcal{D})$ and $\mathcal{D}^{i+}(\subseteq \mathcal{D})$ be the optimum cover of the points in $\mathcal{P}_{e^{i-}}$ and $\mathcal{P}_{e^{i+}}$ respectively.

Corollary 3.2.7. $|\mathcal{D}^{i-} \cap \mathcal{D}^{i+} \cap \mathcal{L}| \leq 2$.

Proof. Proof of the Corollary follows from Lemma 3.2.6. □

Lemma 3.2.8. $|\mathcal{D}^{i-} \cap \mathcal{D}^{i+} \cap \mathcal{U}| \leq 1$.

Proof. Since the centers of all the disks in $\mathcal{U}$ are in ℓ^+ and the points in $\mathcal{P}$ are in ℓ^-, two disks d_x, d_y in $\mathcal{U}$ cannot intersect twice in ℓ^-. Therefore, $|\mathcal{D}^{i-} \cap \mathcal{D}^{i+} \cap \mathcal{U}|$ is at most 1. Thus, the lemma follows. □

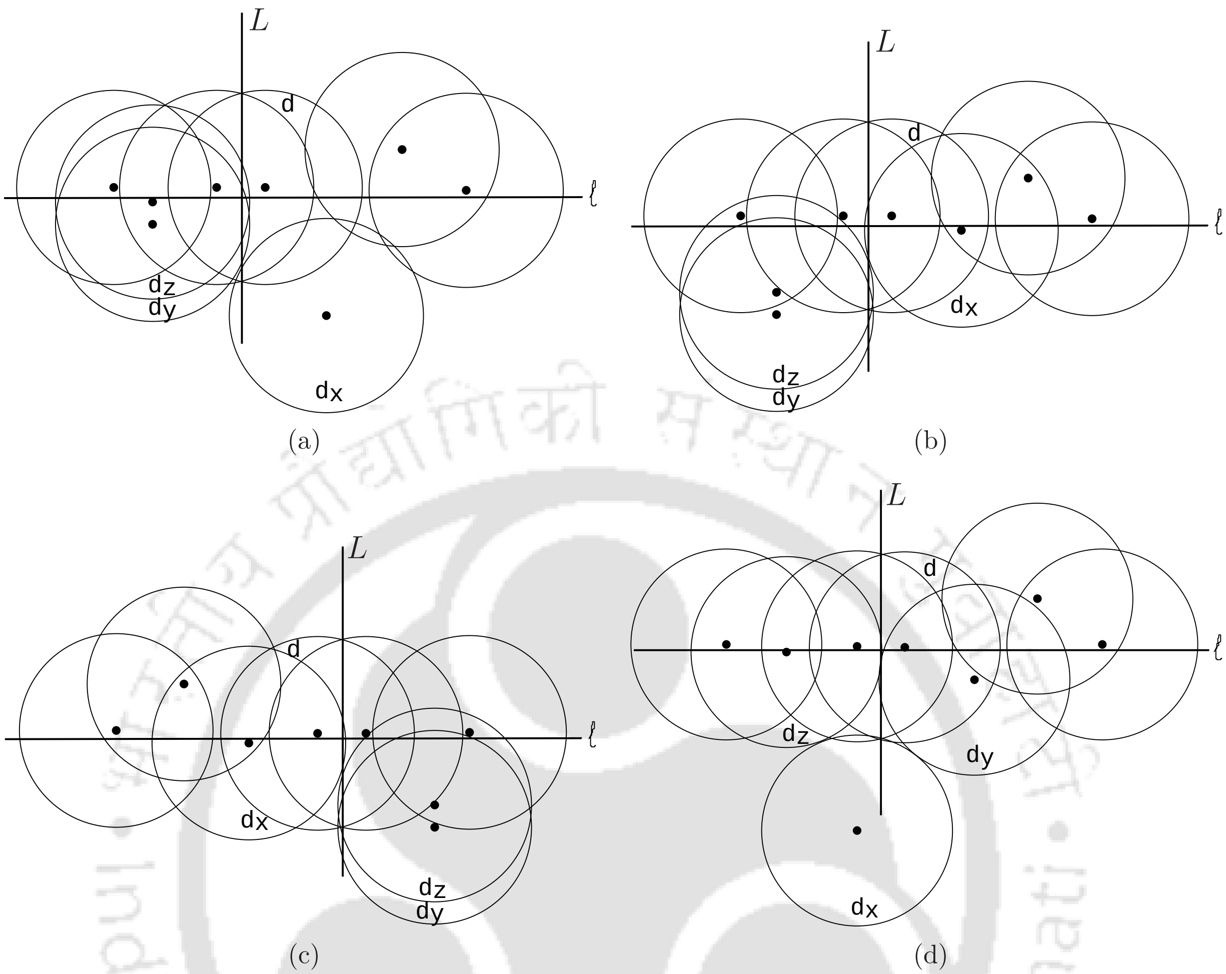


Figure 3.7: Proof of Lemma 3.2.6 (mirror cases)

Lemma 3.2.9. $|\mathcal{D}^{i-} \cap \mathcal{D}^{i+}| \leq 3$.

Proof. Follows from (i) $\mathcal{D} = \mathcal{U} \cup \mathcal{L}$, (ii) Corollary 3.2.7, and (iii) Lemma 3.2.8. $\square$

The following theorem says that the LSDUDC problem admits a PTAS.

Theorem 3.2.10. *Algorithm 3.1 produces* $(1 + \frac{3}{k-3})$*-factor approximation results in* $O(m^k n \log n)$ *time.*

Proof. For some integer t, let $j_1, j_2, \ldots, j_t$ be the values of j in the while loop (line number 9) of the Algorithm 3.1. Let $\mathcal{Q}_v = \bigcup_{i=j_{v-1}+1, j_{v-1}+2, \ldots, j_v} \mathcal{P}_i$ for $v = 1, 2, \ldots, t$, where $j_0 = 0$. Algorithm 3.1 finds a covering for the sets $\{\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_t\}$ independently with each of size k (optimum size because in each iteration of the while loop in line number 9,

Algorithm 3.1 LSDUDC($\mathcal{P}, \mathcal{D}, k, \ell$)

Input: Set $\mathcal{P}$ of points, set $\mathcal{D}$ of unit disks, a positive integer k and a horizontal line ℓ such that $\mathcal{P} \cap \ell^- = \mathcal{P}$ and the union of the disks centered in ℓ^+ covers all the points in $\mathcal{P}$.
Output: Set $\mathcal{D}^* \subseteq \mathcal{D}$ of disks covering all the points in $\mathcal{P}$.
Set $\mathcal{D}^* \leftarrow \emptyset$
Find lower boundary disks set $\mathcal{D}_\ell$ and arrange them from left to right as defined above (see Figure 3.1). Let $\mathcal{D}_\ell = \{d_1, d_2, \ldots, d_s\}$ be the lower boundary disks from left to right.
for ($i = 1, 2, \ldots s$) **do**
    Compute the set $\mathcal{P}_i (\subseteq \mathcal{P})$
end for
$i \leftarrow 1$
while ($i \leq s$) **do**
    Find the maximum index j such that $\bigcup_{t=i,i+1,\ldots j} \mathcal{P}_t$ is covered by a set $\mathcal{D}_1 (\subseteq \mathcal{D})$ of disks with $k = |\mathcal{D}_1|$.
    $\mathcal{D}^* = \mathcal{D}^* \cup \mathcal{D}_1$, $i \leftarrow j + 1$
end while
Return $\mathcal{D}^*$

Algorithm 3.1 finds maximum value of j's) except the covering of $\mathcal{Q}_t$. Let $\mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^t$ be the covering for $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_t$ respectively.

Consider any vertical line e^i. Let $\widehat{\mathcal{D}} = \mathcal{D}' \cup \mathcal{D}''$ be the optimal cover for the point set $\mathcal{P}$, where $\mathcal{D}'$ is a minimum cardinality subset of $\widehat{\mathcal{D}}$ covering all points in $\mathcal{P}_{e^{i-}}$ and $\mathcal{D}''$ is a minimum cardinality subset of $\widehat{\mathcal{D}}$ covering all points in $\mathcal{P}_{e^{i+}}$. Let $|\mathcal{D}' \cap \mathcal{D}''| = r$ (≥ 0). By Lemma 3.2.6 and Lemma 3.2.8, we can replace r disks of $\mathcal{D}' \cap \mathcal{D}''$ in $\mathcal{D}'$ with at most three disks and we can still cover $\mathcal{P}_{e^{i-}}$. Therefore, $|\mathcal{D}^{i-}| \leq |\mathcal{D}'| - r + 3$. By the minimum cardinality of $\mathcal{D}^{i+}$, $|\mathcal{D}^{i+}| \leq |\mathcal{D}''|$. Then,

$$\begin{aligned} |\mathcal{D}^{i-}| + |\mathcal{D}^{i+}| &\leq |\mathcal{D}'| + |\mathcal{D}''| - r + 3 \\ \implies |\mathcal{D}^{i-}| + |\mathcal{D}^{i+}| - 3 &\leq |\mathcal{D}'| + |\mathcal{D}''| - r \\ \implies |\mathcal{D}^{i-}| + |\mathcal{D}^{i+}| - 3 &\leq |\widehat{\mathcal{D}}| \\ \implies |\mathcal{D}^{i-} \cup \mathcal{D}^{i+}| - 3 &\leq |\widehat{\mathcal{D}}| \end{aligned} \tag{3.2.1}$$

Algorithm 3.1 computed optimal covers $\mathcal{D}^i$ (of size k) for $\mathcal{Q}_i$ independently for every

$i = 1, 2, \ldots, t-1$. Hence, for every $i = 1, 2, \ldots, t-1$, let $\mathcal{D}^i \cap \mathcal{D}^{i+1} = \emptyset$ in the worst case, whereas $|\mathcal{D}^{i-} \cap \mathcal{D}^{i+}| \leq 3$ by Lemma 3.2.9. Then, from inequality (3.2.1) the lower bound on the size of optimal solution for the point set $\mathcal{P}$ is $|\bigcup_{i=1,2,\ldots,t-1} \mathcal{D}^i| + |\mathcal{D}^t| - 3(t-1) = k(t-1) + |\mathcal{D}^t| - 3(t-1) = (k-3)(t-1) + |\mathcal{D}^t|$. Therefore, the total number of disks required to cover all the points by Algorithm 3.1 is $k(t-1) + |\mathcal{D}^t|$ whereas at least $(k-3)(t-1) + |\mathcal{D}^t|$ disks required in the optimum solution. Thus, the approximation factor of the Algorithm 3.1 is $(1 + \frac{3}{k-3})$.

The execution time to find lower boundary disks and to arrange them from left to right (line number 4) is $O(m \log m)$, where $m = |\mathcal{D}|$. To compute $\mathcal{P}_i$ for $i = 1, 2, \ldots s$ (for loop at line number 5) $O(n \log n)$ time is required, where $n = |\mathcal{P}|$. To implement the while loop (line number 9), we first create set $\mathcal{P}^u (\subseteq \mathcal{P})$ of points such that $\mathcal{P}^u = \bigcup_{i=1,2,\ldots,u} \mathcal{P}_i$ for each $u = 1, 2, \ldots, s$, then for maximum j, we choose $j = 2^i$ for $i = 1, 2, \ldots, v$ such that $\mathcal{P}^{2^v}$ is not covered by k disks but $\mathcal{P}^{2^{v-1}}$ is covered by k disks. Now, we need to perform a binary search among $[2^{v-1}+1, 2^{v-1}+2, \ldots, 2^v]$ for the maximum value of j. Therefore, the time complexity of the while loop (line number 9) is $O(m^k n \log n)$. Thus, the total time complexity of the Algorithm 3.1 is $O(m^k n \log n)$. □

3.2.1 $(9 + \epsilon)$-Approximation Algorithm for DUDC Problem

In this section we wish to describe a $(9 + \epsilon)$-approximation algorithm for the DUDC problem. Here a set $\mathcal{P}$ of n points and a set $\mathcal{D}$ of m unit disks are distributed in the plane; the objective is to (i) check whether the union of all the disks in $\mathcal{D}$ covers all the points in $\mathcal{P}$, and (ii) if so, then choose a minimum cardinality set $\mathcal{D}^* (\subseteq \mathcal{D})$ such that the union of the disks in $\mathcal{D}^*$ covers $\mathcal{P}$. Since checking the feasibility of the DUDC problem is simple using the method described in Section 3.1, we always assume that the given instance of the DUDC problem is feasible. From Theorem 3.2.10, the LSDUDC problem has an $(1 + \mu)$-approximation algorithm ($\mu = \frac{3}{k-3}$) and the running time of the algorithm is $O(m^{3(1+\frac{1}{\mu})} n \log n)$. Das et al. [23] proposed an approximation algorithm for the DUDC problem using algorithms for the LSDUDC and the WSDUDC (with strip height $1/\sqrt{2}$) problems (see Section 3.1) and proved that the approximation factor of the algorithm for the DUDC problem is

6× (approximation factor of an algorithm for the LSDUDC problem) + approximation factor of an algorithm for the WSDUDC (height $\delta = 1/\sqrt{2}$) problem.

Fraser and López-Ortiz [33] proposed a 3-approximation algorithm for the WSDUDC (with height $\delta \leq 4/5$) problem in $O(m^6 n)$ time. Therefore, we have the following theorem for the DUDC problem.

Theorem 3.2.11. *The DUDC problem has a $(9+\epsilon)$-approximation algorithm with running time $O(m^{3(1+\frac{6}{\epsilon})} n \log n)$.*

Proof. From Lemma 3.1.1, 3.1.2 and Theorem 3.1.3, the approximation factor of the algorithm for the DUDC problem is (6× (approximation factor of an algorithm for the LSDUDC problem) + approximation factor of an algorithm for the WSDUDC (height $= 1/\sqrt{2}$) problem) [23]. Therefore, the approximation factor of the algorithm for the DUDC problem is $6 \times (1+\mu) + 3 = 9 + \epsilon$, where $\epsilon = 6\mu$ (see Theorem 3.2.10 and Section 3.1). The time complexity follows from the time complexity of the WSDUDC [33], and the complexity result stated in Theorem 3.2.10. □

3.3 Approximation Algorithms for the RRC Problem

In the RRC problem, the inputs are a set $\mathcal{D}$ of m unit disks and a rectangular region $\mathcal{R}$; the objective is (i) to check whether the union of all the disks in $\mathcal{D}$ covers $\mathcal{R}$, and (ii) if so, then choose a minimum cardinality set $\mathcal{D}^{**} \subseteq \mathcal{D}$ such that $\mathcal{R} \subseteq \bigcup_{d \in \mathcal{D}^{**}} d$.

Given an instance $(\mathcal{D}, \mathcal{R})$ of the RRC problem, the feasibility checking procedure for that instance is as follows:

- Let $\mathcal{Q}$ be the set of centers of unit disks in $\mathcal{D}$. Draw the nearest-point Voronoi diagram $VOR(\mathcal{Q})$ on the point set $\mathcal{Q}$ within the rectangular region $\mathcal{R}$, where $vor(q_i)$ denotes the Voronoi cell corresponding to the point $q_i \in \mathcal{Q}$ for $i = 1, 2, \ldots, m$.

- Let d_i be the unit disk centered at q_i for $i = 1, 2, \ldots, m$. For each Voronoi cell $vor(q_i)$ if $vor(q_i) \subseteq d_i$, then the given instance $(\mathcal{D}, \mathcal{R})$ for the RRC problem has a feasible solution, otherwise the given instance has no feasible solution.

To check whether the Voronoi cell $vor(q_i)$ is covered by the corresponding unit disk d_i ($i = 1, 2, \ldots, m$), it is sufficient to check whether every vertex on the boundary of $vor(q_i)$ lies inside d_i or not. The number of vertices on the boundary of $vor(q_i)$ is the same as the number of edges on the boundary of $vor(q_i)$. The number of edges in $VOR(\mathcal{Q})$ is $O(m)$. Since each edge lies on the boundary of at most two Voronoi cells $vor(q_i)$ and $vor(q_j)$ for $i \neq j$, checking whether $vor(q_i) \subseteq d_i$ or not takes $O(m)$ time. Therefore the running time of the feasibility checking procedure is dominated by the time required to compute the Voronoi diagram. Hence, the feasibility checking procedure runs in $O(m \log m)$ time. Since feasibility checking for the RRC problem is simple, from now on we assume that a given instance of the RRC problem has a feasible solution.

A *sector* f inside $\mathcal{R}$ is a maximal region inside $\mathcal{R}$ formed by the intersection of a set of disks. Thus each point within f is covered by the same set of disks. Let $\mathcal{F}$ be the set of all sectors (inside $\mathcal{R}$) formed by $\mathcal{D}$, and $|\mathcal{F}| = O(m^2)$. Now we construct a set of points $\mathcal{T}$ as follows: for each sector $f \in \mathcal{F}$ we add one arbitrary point $p \in f$ to $\mathcal{T}$. Therefore, covering all the sectors in $\mathcal{F}$ by a minimum cardinality subset of $\mathcal{D}$ is equivalent to covering all the points in $\mathcal{T}$ by the same subset of $\mathcal{D}$. Thus, we have the following theorem:

Theorem 3.3.1. *The RRC problem has a* $(9+\epsilon)$*-approximation algorithm with running time* $O(m^{5+\frac{18}{\epsilon}} \log m)$.

Proof. Consider an arbitrary point $p \in \mathcal{T}$. Let $f \in \mathcal{F}$ be the sector in which the point p lies. From the definition of sector, if a disk $d \in \mathcal{D}$ covers p, then the disk d also covers the whole sector f. Therefore, the instance $(\mathcal{D}, \mathcal{R})$ of the RRC problem is exactly same as the instance $(\mathcal{T}, \mathcal{D})$ of the DUDC problem. Note that $|\mathcal{T}| = O(m^2)$. Thus, the theorem follows from Theorem 3.2.11 by putting $n = m^2$. □

3.3.1 RRC Problem in Reduce Radius Setup

In this subsection we consider the RRC problem in reduce radius setup. In this setup, a set $\mathcal{D}$ of unit disks and a rectangular region $\mathcal{R}$ such that $\mathcal{R}$ is covered by the union of the disks in $\mathcal{D}$ after reducing their radius to $(1-\gamma)$ are given. The objective is to choose a minimum cardinality set $\mathcal{D}^{**}(\subseteq \mathcal{D})$ whose union covers $\mathcal{R}$. In the reduce radius setup an algorithm $\mathcal{A}$ is said to be a β-approximation if $\frac{|\mathcal{A}_{out}|}{|opt|} \leq \beta$, where $\mathcal{A}_{out}$ is the output of $\mathcal{A}$ and *opt* is the optimum set of disks with reduced radius covering the region of interest. Reduce radius setup has many applications in wireless sensor networks, where coverage remains stable under small perturbations of sensing ranges and their positions. Here, we propose a 2.25-approximation algorithm for this problem. The best known approximation factor for the same problem was 4 [32].

Claim 3.3.2. Let $\nu = \sqrt{2}\gamma$ and d be an unit disk centered at a point p. If d' is a disk of radius $(1-\gamma)$ centered within a square $\mathcal{S}$ of size $\nu \times \nu$ centered at p, then $d' \subseteq d$.

Proof. Let c be the length of the diagonal of $\mathcal{S}$. Then, the maximum distance of any point within the square $\mathcal{S}$ of size $\nu \times \nu$ from the center point p is $c/2 = \gamma$. Thus, the Claim follows. □

Consider a grid with cells of size $\nu \times \nu$ over the region $\mathcal{R}$. Like Funke et al. [32] we also snap the center of each $d \in \mathcal{D}$ to the closest vertex of the grid and set its radius to $(1-\gamma)$. Let $\mathcal{D}'$ be the set of disks with radius $(1-\gamma)$ after snapping their centers. Let $\mathcal{R}'$ be a square of size $2l \times 2l$ on the plane contained in $\mathcal{R}$, where l is a positive integer. We define the regions *UP, DOWN, LEFT, RIGHT, UP-LEFT, UP-RIGHT, DOWN-LEFT, DOWN-RIGHT* around $\mathcal{R}'$ as shown in Figure 3.8. We now construct a set $\mathcal{D}_{RS}(\subseteq \mathcal{D}')$ such that any disk $d \in \mathcal{D}'$ and $d \notin \mathcal{D}_{RS}$ cannot participate in the optimum solution (minimum size) for covering the region $\mathcal{R}'$ by the disks in $\mathcal{D}'$. Note that, if a disk $d \in \mathcal{D}_{RS}$, then the center of d is a grid vertex. The pseudo code for construction of $\mathcal{D}_{RS}$ is given in Algorithm 3.2.

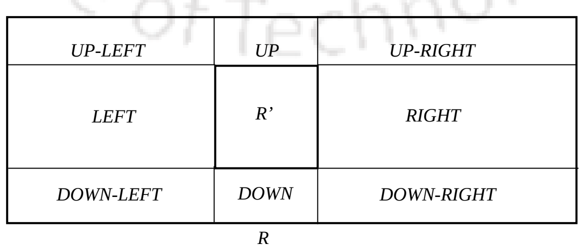


Figure 3.8: Definition of different regions

Definition 3.3.3. A disk $d \in \mathcal{D}'$ dominates another disk $d' \in \mathcal{D}'$ with respect to the region $\mathcal{R}'$ if $d \cap \mathcal{R}' \supseteq d' \cap \mathcal{R}'$.

Algorithm 3.2 *Algorithm_*$\mathcal{D}_{RS}(\mathcal{D}', \mathcal{R}', \nu)$

1: **Input:** Set $\mathcal{D}'$ of disks, a square region $\mathcal{R}'$ of size $2l \times 2l$ and grid size ν.
2: **Output:** $\mathcal{D}_{RS}(\subseteq \mathcal{D}')$ such that no disk $d \notin \mathcal{D}_{RS}$ can participate in the optimum solution for covering the region $\mathcal{R}'$ by $\mathcal{D}'$.
3: Set $\mathcal{D}_{RS} \leftarrow \emptyset$
4: For each disk $d \in \mathcal{D}'$ having center in $\mathcal{R}'$, $\mathcal{D}_{RS} = \mathcal{D}_{RS} \cup \{d\}$
5: For each horizontal grid line segment h in *LEFT*, add a disk $d \in \mathcal{D}'$ to $\mathcal{D}_{RS}$ if (i) $d \cap \mathcal{R}' \neq \emptyset$, (ii) the center of d lies on h and (iii) the center of d is closer to $\mathcal{R}'$ than other disks having centers on h. Similarly add disks to $\mathcal{D}_{RS}$ for the regions *RIGHT*, *UP* and *DOWN*.
6: **for** (each horizontal grid line segment h in *UP-RIGHT* from bottom to top) **do**
7:     Add a disk $d \in \mathcal{D}'$ to $\mathcal{D}_{RS}$ if (i) $d \cap \mathcal{R}' \neq \emptyset$, (ii) the center of d lies on h and (iii) there does not exist any disk $d' \in \mathcal{D}_{RS}$ dominating d.
8: **end for**
9: repeat steps 6-8 for *UP-LEFT*, *DOWN-LEFT* and *DOWN-RIGHT*.
10: Return $\mathcal{D}_{RS}$

Lemma 3.3.4. *If $d \in \mathcal{D}'$ and $d \notin \mathcal{D}_{RS}$, then d cannot participate in the optimum solution for covering $\mathcal{R}'$ by minimum number of disks in $\mathcal{D}'$.*

Proof. The center of d is outside of $\mathcal{R}'$ as $d \notin \mathcal{D}_{RS}$ (see line number 4 of Algorithm 3.2). Without loss of generality assume that the center of d is in *LEFT* and on the horizontal grid line segment h. By our construction of the set $\mathcal{D}_{RS}$, there exists a disk $d' \in \mathcal{D}_{RS}$ centered on h such that (a) $d' \cap \mathcal{R}' \neq \emptyset$, (b) the center of d' lies on h and (c) the center of d' closer to $\mathcal{R}'$ than other disks having centers on h. Therefore, d' dominates d. Similarly, we can prove for other cases also. Thus, the lemma follows. □

Lemma 3.3.5. $|\mathcal{D}_{RS}| \leq \lceil \frac{4l^2}{\nu^2} + \frac{8l+4}{\nu} \rceil$.

Proof. The lemma follows from the following facts: (i) the maximum number of grid vertices in $\mathcal{R}'$ is $\frac{4l^2}{\nu^2}$ and each of them can contribute one disk in $\mathcal{D}_{RS}$, (ii) the maximum number of horizontal grid line segments in the regions *UP-LEFT, LEFT, DOWN-LEFT, DOWN-RIGHT, RIGHT* and *UP-RIGHT* that can contribute a disk in $\mathcal{D}_{RS}$ is $\frac{4l+4}{\nu}$ and (iii) the maximum number of vertical grid line segments in the regions *UP* and *DOWN* that can contribute a disk in $\mathcal{D}_{RS}$ is $\frac{4l}{\nu}$. Thus, the lemma follows. □

From Claim 3.3.2 and Lemma 3.3.5, we can compute a cover of $\mathcal{R}'$ by $\mathcal{D}''(\subseteq \mathcal{D})$ with the minimum number of disks using a brute-force method, where $\mathcal{D}''$ is the set of unit disks corresponding to the reduced-radius disks in $\mathcal{D}_{RS}$. The running time of the brute-force algorithm is $O(2^{\lceil \frac{4l^2}{\nu^2}+\frac{8l+4}{\nu} \rceil})$ (see Lemma 3.3.5). Although this worst-case running time of the brute-force algorithm is exponential in $(\frac{l}{\nu})^2$, but it is very small for practical input data. We now describe an approximation factor of our proposed algorithm for RRC problem in reduce radius setup.

Theorem 3.3.6. *In the reduce radius setup, the RRC problem has an $(1+\frac{1}{l})^2$-approximation algorithm with running time $O(ql^2 2^{\lceil \frac{4l^2}{\nu^2}+\frac{8l+4}{\nu} \rceil})$, where q is the minimum number of squares of size $2l \times 2l$ covering $\mathcal{R}$ and l is a positive integer.*

Proof. From the above discussion, for rectangle of size $2l \times 2l$, we have an optimum solution for the RRC problem. Note that the diameter of each disk of the RRC instance is 2. Therefore, by the shifting strategy described by Hochbaum and Maass [50] along horizontal and then vertical directions we have a $(1+1/l)^2$-approximation algorithm for solving the RRC problem in time $O(ql^2 2^{\lceil \frac{4l^2}{\nu^2}+\frac{8l+4}{\nu} \rceil})$. Thus, the theorem follows. □

Corollary 3.3.7. *In the reduce radius setup, the RRC problem has a 2.25-approximation algorithm with running time $O(q2^{\lceil \frac{16}{\nu^2}+\frac{20}{\nu} \rceil})$, where q is the minimum number of squares of size 4×4 covering $\mathcal{R}$.*

Proof. Follows from Theorem 3.3.6 by setting $l = 2$. □

Note that, Funke et al. [32] proposed a 4-approximation algorithm in $O(q2^{\lceil \frac{20}{\nu^2} \rceil})$ time for the RRC problem in reduce radius setup. Thus, our proposed algorithm is a significant improvement over the best known existing algorithm for the same problem.

3.4 Conclusion

In this chapter we have proposed a PTAS for the LSDUDC problem. The running time of our proposed PTAS is $O(m^{3(1+\frac{1}{\mu})} n \log n)$, where $0 < \mu \leq 1$. Using this PTAS, we proposed a $(9+\epsilon)$-approximation algorithm for the DUDC problem, improving previous 15-approximation result for the same problem [33], where $0 < \epsilon \leq 6$. We have also proposed a $(9+\epsilon)$-approximation algorithm for the RRC problem, which runs in $O(m^{5+\frac{18}{\epsilon}} \log m)$ time. In the reduce radius setup, we proposed a PTAS and using this result, we proposed a 2.25-approximation algorithm. The previous best known approximation factor was 4 [32]. The running time of our proposed algorithm for the RRC problem in reduce radius setup is less than that of 4-approximation algorithm proposed in [32] for reasonably large value of $\gamma (= \frac{\nu}{\sqrt{2}})$, where γ is the radius reduction parameter.

Chapter 4

Discrete Unit Square Cover Problem

In this chapter we consider the *discrete unit square cover* (DUSC) problem as follows:

> Given a set $\mathcal{P}$ of n points and a set $\mathcal{S}$ of m axis-aligned unit squares in $\mathbb{R}^2$, the objective is (i) to check whether the union of the squares in $\mathcal{S}$ covers all the points in $\mathcal{P}$, and (ii) if the answer is yes, then select a minimum cardinality subset $\mathcal{S}^* \subseteq \mathcal{S}$ such that each point in $\mathcal{P}$ is covered by at least one square in $\mathcal{S}^*$.

The DUSC problem has been well studied in the literature. All the previous algorithms for the DUSC problem take huge amount of time and are based on complicated techniques [17, 66, 78, 62]. In this chapter, we propose simple approximation algorithms which run faster than the previous approximation algorithms for a certain range of approximation factor.

For the DUSC problem;

(i) We propose a $(2 + \frac{4}{k-2})$-approximation algorithm, where $k(> 2)$ is an integer parameter that defines a trade-off between the running time and the approximation factor of the algorithm. The running time of our proposed algorithm is $O(km^k n)$. Our solution of the *discrete unit square cover* problem is based on a simple $(1 + \frac{2}{k-2})$-approximation algorithm for the subproblem *strip square cover* problem. In the *strip square cover* problem, all the points in $\mathcal{P}$ are lying within a horizontal strip of unit height.

(ii) We also propose a 2-approximation algorithm, which runs in $O(m^4 n + n \log n)$ time. The 2-approximation algorithm is based on an algorithm for the *strip square cover* subproblem. The algorithm for the subproblem is developed using plane sweep and graph search traversal techniques.

In Section 4.1 we describe a procedure for checking the feasibility of an instance of the *discrete unit square cover* problem. We present an $(1 + \frac{2}{k-2})$-approximation algorithm for the *strip square cover* problem in Section 4.2. In Section 4.3, we propose a $(2 + \frac{4}{k-2})$-approximation algorithm for the DUSC problem using an $(1 + \frac{2}{k-2})$-approximation algorithm for the *strip square cover* problem. In Section 4.4, we also propose a 2-approximation algorithm with running time $O(m^4 n + n \log n)$ for the DUSC problem using the similar kind of technique proposed in [78]. Finally, we conclude the chapter in Section 4.5.

4.1 Testing Feasibility of the Discrete Unit Square Cover Problem

We use a plane sweep technique [7] to check the feasibility of an instance of a *discrete unit square cover* problem as follows: imagine sweeping a 1-dimensional vertical line (sweep line) ℓ across the plane from left to right. For each square $s \in \mathcal{S}$ the coordinates $x(s) - 1/2$ and $x(s) + 1/2$ together with the x-coordinates of all the points in $\mathcal{P}$ are the event points, where $x(s)$ (resp. $y(s)$) is the x-coordinate (resp. y-coordinate) of the center[1] of the square s. Therefore, the total number of event points is $2m+n = O(m+n)$. The event queue $\mathcal{Q}$ consists of all event points sorted in the order of their increasing x-coordinates. The sweep line status $\mathcal{T}$ consists of squares s (coordinate $y(s) - 1/2$ of square s) in $\mathcal{S}$ that intersect the sweep line ℓ at its current position. We use the balanced binary search tree as a data structure to maintain the sweep line status $\mathcal{T}$ dynamically. Hence, insertion, deletion and search operations can be performed on $\mathcal{T}$ in $O(\log m)$ time. Let $x(p)$ (resp. $y(p)$) be the x-coordinate (resp. y-coordinate) of the point $p \in \mathcal{P}$. Our plane sweep algorithm for feasibility checking is as follows:

[1] Center of a square s is the intersection point of the two diagonals of s.

1) We simulate sweeping a vertical line ℓ over the plane from left to right.

2) For the current position of the sweep line ℓ, we do the following:

   a) If the next event point in the queue $\mathcal{Q}$ is the point $p(\in \mathcal{P})$, then we search for the square s in the sweep line status $\mathcal{T}$ that has largest y-coordinate but $y(s) - 1/2 < y(p)$. If p is covered by s, we remove it from the queue $\mathcal{Q}$, otherwise we report that the instance of the DUSC problem has no feasible solution.

   b) If the next event point in the queue $\mathcal{Q}$ is a coordinate $x(s) - 1/2$ of a square s, then we insert the square s (its coordinate $y(s) - 1/2$) into the sweep line status $\mathcal{T}$, update the status $\mathcal{T}$ and remove the event point from $\mathcal{Q}$.

   c) If the next event point in the queue $\mathcal{Q}$ is a coordinate $x(s) + 1/2$ of a square s, then we delete the square s (its coordinate $y(s) - 1/2$) from the sweep line status $\mathcal{T}$, update the status $\mathcal{T}$ and remove the event point from $\mathcal{Q}$.

The total running time of the above plane sweep algorithm is $O((m+n)\log m)$. Since feasibility checking is simple, from now onward we consider every instance of the DUSC problem has a feasible solution.

4.2 Strip Square Cover Problem

In this section we consider the *Strip Square Cover* (SSC) problem. Let $\mathcal{H} = [\ell_1, \ell_2]$ be a horizontal strip of height 1, where ℓ_1 and ℓ_2 are upper and lower horizontal lines respectively (see Figure 4.1). The definition of the SSC problem is as follows:

> Given a set $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ of n points lying within an horizontal strip $\mathcal{H}$ and a set $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$ of m unit squares. The objective is (i) to check whether $\mathcal{P} \subset \bigcup_{s\in\mathcal{S}} s$ or not, and (ii) if $\mathcal{P} \subset \bigcup_{s\in\mathcal{S}} s$, then compute a minimum cardinality subset $\mathcal{S}^* \subseteq \mathcal{S}$ such that the union of the squares in $\mathcal{S}^*$ covers all the points in $\mathcal{P}$, i.e., $\mathcal{P} \subseteq \bigcup_{s\in\mathcal{S}^*} s$.

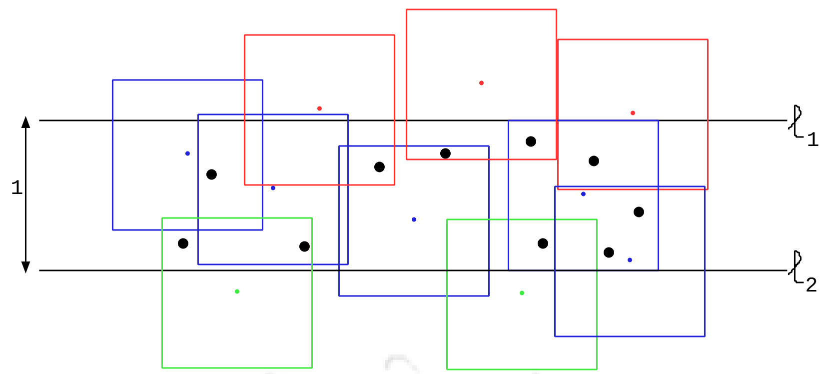


Figure 4.1: An instance of the SSC problem

Note that the SSC problem is a special case of DUSC problem, where the strip is defined by the top-most and bottom-most points and has at most unit height, whereas the DUSC problem has no limit on the height of the strip.

4.2.1 Terminology

We assume that all the points in $\mathcal{P}$ and the centers of squares in $\mathcal{S}$ have distinct x-coordinates. Recall that $x(s)$ (resp. $y(s)$) is the x-coordinate (resp. y-coordinate) of the center of the square $s \in \mathcal{S}$. For the sake of simplicity, let us assume that no square in $\mathcal{S}$ intersects both ℓ_1 and ℓ_2 simultaneously. Let $\mathcal{S}^{\ell_1}$ (resp. $\mathcal{S}^{\ell_2}$) be the set of squares in $\mathcal{S}$ intersecting the upper (resp. lower) horizontal line ℓ_1 (resp. ℓ_2) i.e., $\mathcal{S}^{\ell_1} = \{s \in \mathcal{S} | s \cap \ell_1 \neq \emptyset\}$ and $\mathcal{S}^{\ell_2} = \{s \in \mathcal{S} | s \cap \ell_2 \neq \emptyset\}$. Let L be an arbitrary vertical line. We use L^- (resp. L^+) to denote the region in the left (resp. right) side of the vertical line L i.e., the left (resp. right) half-plane of L. Let $\mathcal{P}^{L-}$ (resp. $\mathcal{P}^{L+}$) be the set of points of $\mathcal{P}$ which lie on L^- (resp. L^+) i.e., $\mathcal{P}^{L-} = \mathcal{P} \cap L^-$ and $\mathcal{P}^{L+} = \mathcal{P} \cap L^+$. Let $\mathcal{S}^{L-}(\subseteq \mathcal{S})$ and $\mathcal{S}^{L+}(\subseteq \mathcal{S})$ be the optimum cover of the points in $\mathcal{P}^{L-}$ and $\mathcal{P}^{L+}$ respectively.

4.2.2 Preliminaries

Claim 4.2.1. If $s_1, s_2 \in \mathcal{S}^{\ell_1}$ and intersect L, then both s_1 and s_2 cannot be in $\mathcal{S}^{L-}$ and $\mathcal{S}^{L+}$ simultaneously i.e, $|(\mathcal{S}^{L-} \cap \mathcal{S}^{L+}) \cap \mathcal{S}^{\ell_1}| \leq 1$.

Proof. On contrary assume that $s_1, s_2 \in \mathcal{S}^{L-}$ and $s_1, s_2 \in \mathcal{S}^{L+}$. Since $s_1, s_2 \in \mathcal{S}^{L-}$ and $s_1, s_2 \in \mathcal{S}^{L+}$, there exist points $p_1 \in \mathcal{P}^{L-}$ and $p_2 \in \mathcal{P}^{L+}$ such that (i) $p_1 \in s_1$ but $p_1 \notin s$ for all $s \in \mathcal{S}^{L-} \setminus \{s_1\}$, and (ii) $p_2 \in s_2$ but $p_2 \notin s$ for all $s \in \mathcal{S}^{L+} \setminus \{s_2\}$. Similarly, there exist points $p_1' \in \mathcal{P}^{L-}$ and $p_2' \in \mathcal{P}^{L+}$ such that (i) $p_1' \in s_2$ but $p_1' \notin s$ for all $s \in \mathcal{S}^{L-} \setminus \{s_2\}$, and (ii) $p_2' \in s_1$ but $p_2' \notin s$ for all $s \in \mathcal{S}^{L+} \setminus \{s_1\}$.

We have the following four cases based on the center positions of s_1 and s_2:

(a) $y(s_2) < y(s_1)$ and $x(s_2) < x(s_1)$

(b) $y(s_2) < y(s_1)$ and $x(s_2) > x(s_1)$

(c) $y(s_2) > y(s_1)$ and $x(s_2) < x(s_1)$

(d) $y(s_2) > y(s_1)$ and $x(s_2) > x(s_1)$

In Case of (a), since s_1 and s_2 are of same size, $s_1 \cap (L^- \cap \mathcal{H}) \subset s_2 \cap (L^- \cap \mathcal{H})$ (see Figure 4.2). Therefore, $s_1 \cap \mathcal{P}^{L-} \subseteq s_2 \cap \mathcal{P}^{L-}$, which contradicts that the point p_1 is covered only by s_1. We can handle other cases in a similar fashion. Thus the claim follows.

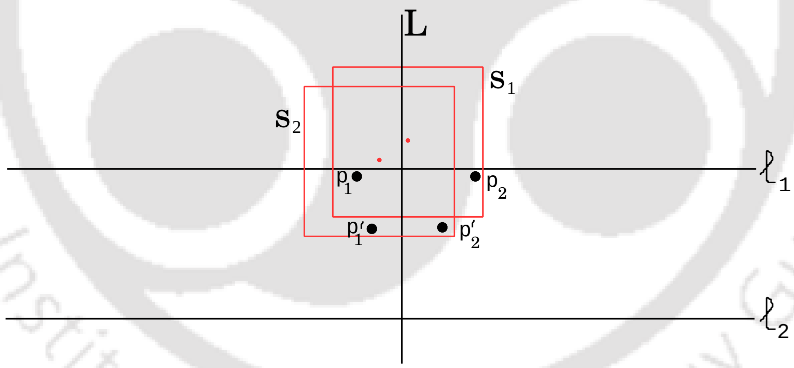


Figure 4.2: Illustration of two squares in $\mathcal{S}^{\ell_1}$ intersecting line L and strip $\mathcal{H} = [\ell_1, \ell_2]$

□

Claim 4.2.2. If $s_1, s_2 \in \mathcal{S}^{\ell_2}$ and intersect L, then both s_1 and s_2 cannot be in $\mathcal{S}^{L-}$ and $\mathcal{S}^{L+}$ simultaneously i.e, $|(\mathcal{S}^{L-} \cap \mathcal{S}^{L+}) \cap \mathcal{S}^{\ell_2}| \leq 1$.

Proof. The claim follows from the same argument as in Claim 4.2.1. For basic idea behind the proof see Figure 4.3. □

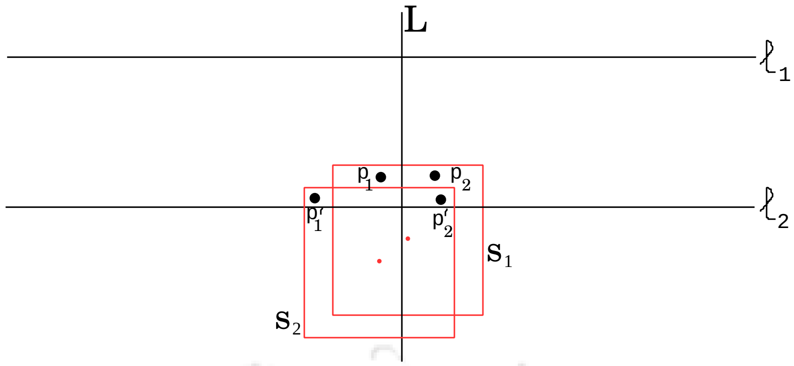


Figure 4.3: Illustration of two squares in $\mathcal{S}^{\ell_2}$ intersecting line L and strip $\mathcal{H} = [\ell_1, \ell_2]$

Lemma 4.2.3. $|\mathcal{S}^{L-} \cap \mathcal{S}^{L+}| \leq 2$.

Proof. Follows from (i) $\mathcal{S} = \mathcal{S}^{\ell_1} \cup \mathcal{S}^{\ell_2}$, (ii) Claim 4.2.1 and Claim 4.2.2. □

4.2.3 Approximation Algorithm for Strip Square Cover Problem

In this subsection we propose an approximation algorithm for the *strip square cover* problem. Here, a set $\mathcal{S}$ of m unit squares and a set $\mathcal{P}$ of n points inside a horizontal strip $\mathcal{H}$ of height 1 are given, the objective is to choose a minimum cardinality subset $\mathcal{S}^* \subseteq \mathcal{S}$ such that $\mathcal{P} \subseteq \bigcup_{s \in \mathcal{S}^*} s$. Our algorithm proceeds as follows: Arrange the points in $\mathcal{P}$ from left to right based on their x-coordinates. Next, apply an exhaustive search on all subsets of size k to choose k squares from $\mathcal{S}$ as members of $\mathcal{S}_{out}$, which cover the maximum number of consecutive points in $\mathcal{P}$ starting from the the left most uncovered point, where $\mathcal{S}_{out}$ is the output of our algorithm (Algorithm 4.1). Continue the above process until all the points in $\mathcal{P}$ are covered. The detailed pseudocode of the algorithm is available in Algorithm 4.1.

Theorem 4.2.4. *Algorithm 4.1 returns a* $(1 + \frac{2}{k-2})$*-approximation result for the strip square cover problem in* $O(km^k n)$ *time, where* $k > 2$.

Proof. In the while loop at line number 5 of Algorithm 4.1, let $\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^t$ be the set of squares computed to cover points in $\mathcal{P}^1, \mathcal{P}^2, \ldots, \mathcal{P}^t$ respectively, where $|\mathcal{S}^i| = k$ for $1 \leq i < t$ and $|\mathcal{S}^t| \leq k$, and $\bigcup_{i=1}^{k} \mathcal{P}^i = \mathcal{P}$. Lemma 4.2.3 says that $|\mathcal{S}^i \cap \mathcal{S}^{i+1}| \leq 2$. Observe

Algorithm 4.1 Strip_Square_Cover($\mathcal{P}, \mathcal{S}, \mathcal{H}, k$)

1: **Input:** A set $\mathcal{P}$ of n points, a set $\mathcal{S}$ of m unit squares, an integer $k > 2$ and a horizontal strip $\mathcal{H}$ of height 1.
2: **Output:** A subset $\mathcal{S}_{out} \subseteq \mathcal{S}$ such that $\mathcal{P} \subseteq \bigcup_{s \in \mathcal{S}_{out}} s$.
3: Set $\mathcal{S}_{out} \leftarrow \emptyset$, $P' \leftarrow \mathcal{P}$
4: Arrange the points in P' from left to right.
5: **while** ($P' \neq \emptyset$) **do**
6:     Find the set $S' \subseteq \mathcal{S}$ of size k such that the union of squares in S' covers the maximum number of consecutive points in P' starting from the left most point.
7:     $P' = P' \setminus ((\bigcup_{s \in S'} s) \cap P')$
8:     $\mathcal{S}_{out} = \mathcal{S}_{out} \cup S'$
9: **end while**
10: Return $\mathcal{S}_{out}$

that $\mathcal{S}^i \cap \mathcal{S}^{i+2} = \emptyset$ for $i = 1, 2, \ldots, t-2$ (see Figure 4.4). Therefore, the lower bound on the size of optimum cover for all points in $\mathcal{P}$ is $\sum_{i=1}^{t} |\mathcal{S}^i| - 2(t-1) = k(t-1) + |\mathcal{S}^t| - 2(t-1) = (k-2)(t-1) + |\mathcal{S}^t|$ i.e., $|\mathcal{S}^*| \geq (k-2)(t-1) + |\mathcal{S}^t|$, where $\mathcal{S}^* \subseteq \mathcal{S}$ is the minimum cardinality set such that $\mathcal{P} \subseteq \bigcup_{s \in \mathcal{S}^*} s$. Algorithm 4.1 outputs a set of squares $\mathcal{S}_{out}$, where $\mathcal{S}_{out} = \mathcal{S}^1 \cup \mathcal{S}^2 \cup \ldots \cup \mathcal{S}^t$ and $|\mathcal{S}^1 \cup \mathcal{S}^2 \cup \ldots \cup \mathcal{S}^t| \leq k(t-1) + |\mathcal{S}^t|$. Therefore,

$$\frac{|\mathcal{S}_{out}|}{|\mathcal{S}^*|} \leq \frac{k(t-1) + |\mathcal{S}^t|}{(k-2)(t-1) + |\mathcal{S}^t|} \leq \frac{k(t-1)}{(k-2)(t-1)} = 1 + \frac{2}{k-2} \tag{4.2.1}$$

Hence, it follows from inequality (4.2.1) that the approximation factor of Algorithm 4.1 is $1 + \frac{2}{k-2}$.

Arranging the points in $\mathcal{P}$ from left to right takes $O(n \log n)$ time. Note that $\mathcal{P}^1, \mathcal{P}^2, \ldots, \mathcal{P}^t$ are the subsets of points in $\mathcal{P}$ that are covered by squares in $\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^t$, respectively. Note that the points in $\mathcal{P}$ are already arranged from left to right. We perform the following operations to compute a set $\mathcal{S}^i$ of size k in the while-loop at line 6 of Algorithm 4.1:

a) We enumerate all $\binom{m}{k}$ subsets of k squares from set $\mathcal{S}$.

b) For each subset of k squares, we mark all the consecutive points of $\mathcal{P}$ starting from the left-most uncovered point, that are covered by these k squares, and remember the subset $\mathcal{S}^i$ of k squares which covers the maximum number of consecutive points

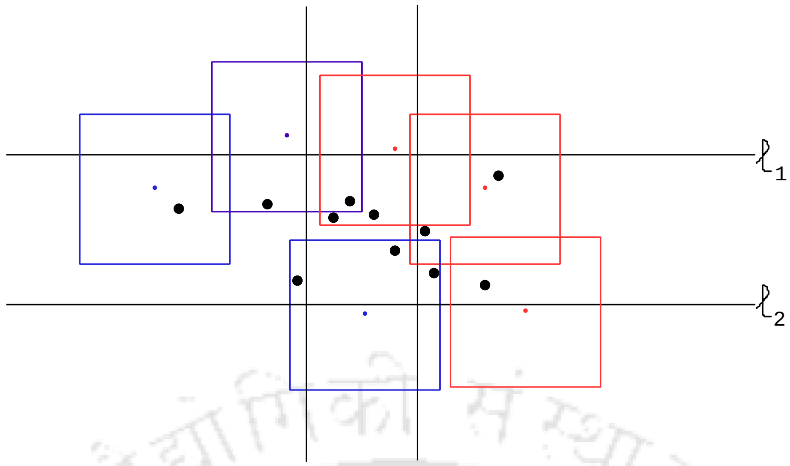


Figure 4.4: Illustration of $\mathcal{S}^i \cap \mathcal{S}^{i+2} = \emptyset$

of $\mathcal{P}$ starting from the left-most uncovered point.

c) We repeat this until all $\binom{m}{k}$ subsets are exhausted.

Then, $k(|\mathcal{P}_1^i|+|\mathcal{P}_2^i|+\ldots+|\mathcal{P}^i_{\binom{m}{k}}|) \leq k\binom{m}{k}|\mathcal{P}^i|$, where $\mathcal{P}^i = \max_{|\mathcal{P}_j^i|}\{\mathcal{P}_j^i | j = 1, 2, \ldots, \binom{m}{k}\}$ and $\mathcal{P}_j^i$ is a set of consecutive points of $\mathcal{P}$ that are covered by some subset of k squares. Hence, the running time of the step in line 6 is $O(m^k k|\mathcal{P}^i|)$. In the while loop at line number 6, computing a set $\mathcal{S}^i$ of size k takes $O(m^k k|\mathcal{P}^i|)$ time. Hence, the total running time of while loop at line number 5 is $O(m^k k|\mathcal{P}^1|+m^k k|\mathcal{P}^2|+\ldots+m^k k|\mathcal{P}^t|) = O(km^k n)$. Since $k > 2$, the overall running time of Algorithm 4.1 is $O(km^k n)$. □

4.3 Approximation Algorithm for the Discrete Unit Square Cover Problem

In this section we consider the *discrete unit square cover* (DUSC) problem in $\mathbb{R}^2$. In the DUSC problem, given a set $\mathcal{P}$ of n points and a set $\mathcal{S}$ of m unit squares in $\mathbb{R}^2$, we wish to cover all the points in $\mathcal{P}$ with the minimum number of squares in $\mathcal{S}$. Let $\mathcal{R}$ be the smallest axis-aligned rectangle containing all points in $\mathcal{P}$ and centers of all squares in $\mathcal{S}$. To cover the points in $\mathcal{P}$ with squares in $\mathcal{S}$, we first partition the rectangle $\mathcal{R}$ into horizontal strips $\mathcal{H}_1 = [\ell_1, \ell_2], \mathcal{H}_2 = [\ell_2, \ell_3], \ldots, \mathcal{H}_r = [\ell_r, \ell_{r+1}]$ of height 1, where ℓ_i and ℓ_{i+1} are the horizontal lines defining the strip $\mathcal{H}_i$, $1 \leq i \leq r$. Next, we invoke Algorithm

Algorithm 4.2 Discrete_Unit_Square_Cover($\mathcal{P}, \mathcal{S}, k$)

1: **Input:** A set $\mathcal{P}$ of n points, a set $\mathcal{S}$ of m unit squares in 2D, and an integer $k(>2)$.

2: **Output:** A subset $\mathcal{S}_{out} \subseteq \mathcal{S}$ such that $\mathcal{P} \subset \bigcup_{s \in \mathcal{S}_{out}} s$.

3: Set $\mathcal{S}_{out} \leftarrow \emptyset$

4: Let $\mathcal{R}$ be the smallest axis-aligned rectangular region containing all points in $\mathcal{P}$ and the centers of all squares in $\mathcal{S}$. Partition $\mathcal{R}$ into horizontal strips $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_r$ of height 1 such that $\bigcup_{i=1}^{r} \mathcal{H}_i \supseteq \mathcal{R}$.

5: **for** $(i = 1, 2 \ldots, r)$ **do**

6: $\quad \mathcal{S}_i = \{s \in \mathcal{S} | \alpha(s) \in \mathcal{H}_i$, where $\alpha(s)$ is the center of square $s\}$

7: **end for**

8: Set $\mathcal{S}_0 \leftarrow \emptyset$, $\mathcal{S}_{r+1} \leftarrow \emptyset$

9: **for** $(i = 1, 2 \ldots, r)$ **do**

10: $\quad \mathcal{P}' = \mathcal{P} \cap \mathcal{H}_i$

11: $\quad \mathcal{S}' = \mathcal{S}_{i-1} \cup \mathcal{S}_i \cup \mathcal{S}_{i+1}$

12: $\quad \mathcal{S}_{out} = \mathcal{S}_{out} \cup$ Strip_Square_Cover($\mathcal{P}', \mathcal{S}', \mathcal{H}_i, k$) /* Call Algorithm 4.1 */

13: **end for**

14: Return $\mathcal{S}_{out}$

4.1 to solve the subproblem restricted to each of these strips $\mathcal{H}_i$ independently. The detailed pseudocode of the algorithm is available in Algorithm 4.2.

Theorem 4.3.1. *Algorithm 4.2 is a* $(2 + \frac{4}{k-2})$*-approximation algorithm for the DUSC problem in* $O(km^k n)$ *time, where* $k(>2)$ *is an integer.*

Proof. In the for-loop at line number 9 of Algorithm 4.2, the height of every horizontal strip $\mathcal{H}_i$ is 1 and all the squares in $\mathcal{S}$ have unit side length. Therefore, any square centered within strip $\mathcal{H}_i$ can participate in the covering of points lying in either strips $\mathcal{H}_{i-1}$ and $\mathcal{H}_i$ only or $\mathcal{H}_{i+1}$ and $\mathcal{H}_i$ only, but not both (see line number 11 of the Algorithm 4.2). Now, let $\mathcal{P}^i = \mathcal{P} \cap \mathcal{H}_i$ for $i = 1, 2, \ldots, r$. Let $\mathcal{S}^i_{out}$ be a set of squares computed in for-loop at line number 9 of Algorithm 4.2 for covering points $\mathcal{P}^i$ lying in strip $\mathcal{H}_i$ for $i = 1, 2, \ldots, r$. Then, let $\mathcal{S}_{out} = \bigcup_{i=1}^{r} \mathcal{S}^i_{out}$ whenever $\mathcal{P}^i$ is non-empty. Let $\mathcal{S}^*$ be an optimal cover of points in $\mathcal{P}$ and $\mathcal{S}^*_i$ be a set of squares from $\mathcal{S}^*$ covering points in $\mathcal{P}^i$ for $i = 1, 2, \ldots, r$. Theorem 4.2.4 says that $|\mathcal{S}^i_{out}| \leq (1 + \frac{2}{k-2})|\mathcal{S}^{**}_i|$ as $\mathcal{S}^i_{out}$ is computed by $(1 + \frac{2}{k-2})$-approximation algorithm, where $\mathcal{S}^{**}_i$ is an optimal cover for points in $\mathcal{P}^i$. Since $\mathcal{S}^*_i$ is a feasible cover for points $\mathcal{P}^i$, $|\mathcal{S}^{**}_i| \leq |\mathcal{S}^*_i|$. Then, $|\mathcal{S}^i_{out}| \leq (1 + \frac{2}{k-2})|\mathcal{S}^*_i|$ and

$\sum_{i=1}^{r} |\mathcal{S}_i^*| \leq 2|\mathcal{S}^*|$ as a square $s \in \mathcal{S}^*$ covering points lying in either (i) $\mathcal{H}_i$ and $\mathcal{H}_{i-1}$ or (ii) $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$, may be counted twice in the summation on the left-hand side. Therefore,

$$|\mathcal{S}_{out}| \leq \sum_{i=1}^{r} |\mathcal{S}_{out}^i| \leq (1 + \frac{2}{k-2}) \sum_{i=1}^{r} |\mathcal{S}_i^*| \leq 2(1 + \frac{2}{k-2})|\mathcal{S}^*|$$

$$\iff |\mathcal{S}_{out}| \leq 2(1 + \frac{2}{k-2})|\mathcal{S}^*| \quad (4.3.1)$$

Hence, it follows from inequality (4.3.1) that the approximation factor of Algorithm 4.2 is $2(1 + \frac{2}{k-2}) = 2 + \frac{4}{k-2}$.

The running time of each call to Algorithm 4.1 at line 12 of Algorithm 4.2 is $O(km_i^k n_i)$ for $i = 1, 2, \ldots, r$, where $m_i \leq m$ and $n_i \leq n$ (see Theorem 4.2.4). Note that $\sum_{i=1}^{r} n_i = n$ and $\sum_{i=1}^{r} m_i \leq 3m$ as every set $\mathcal{S}_i$ of squares is used at most three times at line number 12 in Algorithm 4.2 for covering points in $\mathcal{P}$. Therefore,

$$\sum_{i=1}^{r} km_i^k n_i \leq \sum_{i=1}^{r} km^k n_i = km^k(\sum_{i=1}^{r} n_i) = km^k n \quad (4.3.2)$$

Hence, it follows from (4.3.2) that the overall running time of Algorithm 4.2 is $O(km^k n)$. Thus the theorem follows. □

4.4 2-Approximation Algorithm for the DUSC Problem

In this section we first propose an algorithm to solve the SSC problem optimally in $O(m^4 n + n \log n)$ time. Using the algorithm for the SSC problem, we propose a 2-approximation algorithm for the DUSC problem, which runs in $O(m^4 n + n \log n)$ time.

4.4.1 Algorithm for the Strip Square Cover (SSC) Problem

Recall that the SSC problem is defined as follows: given a set $\mathcal{P}$ of n points lying inside a horizontal strip $\mathcal{H} = [\ell_1, \ell_2]$ of unit height and a set $\mathcal{S}$ of m axis-aligned unit squares such that the union of squares in $\mathcal{S}$ covers all points in $\mathcal{P}$, the aim is to find a minimum cardinality set $\mathcal{S}^* \subseteq \mathcal{S}$ such that the union of squares in $\mathcal{S}^*$ covers all the points in $\mathcal{P}$. Let $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ be the set of n points arranged in an order of increasing x-coordinates. Recall that $\mathcal{S}^{\ell_1}$ (resp. $\mathcal{S}^{\ell_2}$) is the set of squares in $\mathcal{S}$ intersecting the upper (resp. lower) horizontal line ℓ_1 (resp. ℓ_2) i.e., $\mathcal{S}^{\ell_1} = \{s \in \mathcal{S} | s \cap \ell_1 \neq \emptyset\}$ and $\mathcal{S}^{\ell_2} = \{s \in \mathcal{S} | s \cap \ell_2 \neq \emptyset\}$. Let s_1^l, s_1^r, s_2^l and s_2^r be the additional four dummy squares having no intersection with any square in $\mathcal{S}$ such that both s_1^l and s_1^r intersect ℓ_1, both s_2^l and s_2^r intersect ℓ_2, both s_1^l and s_2^l lie to the left of all squares in $\mathcal{S}$, and both s_1^r and s_2^r lie to the right of all squares in $\mathcal{S}$. Let $\mathcal{S} = \mathcal{S} \cup \{s_1^l, s_1^r, s_2^l, s_2^r\}$, $\mathcal{S}^{\ell_1} = \mathcal{S}^{\ell_1} \cup \{s_1^l, s_1^r\}$ and $\mathcal{S}^{\ell_2} = \mathcal{S}^{\ell_2} \cup \{s_2^l, s_2^r\}$. Now, we propose an algorithm based on the plane sweep technique and graph traversal method to compute an optimal solution for the SSC problem.

We now consider all *triplets* $\mathcal{T}$ defined as follows:

For every *triplet* $T \in \mathcal{T}$, $T = (u, v, x)$ satisfies the following properties:

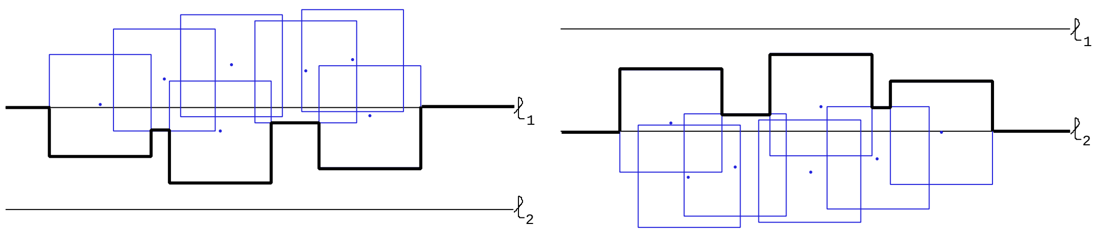
- $u \in \mathcal{S}^{\ell_1}$.
- $v \in \mathcal{S}^{\ell_2}$.
- $x = \max(x(u) - \frac{1}{2},\ x(v) - \frac{1}{2},\ \alpha)$, where α is defined as follows: Initially set $\alpha = x(u) - \frac{1}{2}$. Let $\mathcal{S}_u^{\ell_1} = \{s \in \mathcal{S}^{\ell_1} | x(s) < x(u)\}$ and $\mathcal{S}_v^{\ell_2} = \{s \in \mathcal{S}^{\ell_2} | x(s) < x(v)\}$. Consider a square $s' \in \mathcal{S}_u^{\ell_1} \cup \mathcal{S}_v^{\ell_2}$. If (i) $s' \in \mathcal{S}_u^{\ell_1}$ and $y(s') < y(u)$ or (ii) $s' \in \mathcal{S}_v^{\ell_2}$ and $y(s') > y(v)$ then reset $\alpha = x(s') + \frac{1}{2}$.

Definition 1. Lower envelope

Let $\chi \subseteq \mathcal{S}^{\ell_1}$ be an arbitrary subset. The *Lower envelope* of χ is the union of line segments such that each line segment is either (i) part of ℓ_1 which is not covered by any square in χ or (ii) part of boundary of the area $(\underset{s \in \chi}{\cup} s) \cap \mathcal{H}$ where every point on the boundary lies below ℓ_1 (see thick lines in Figure 4.5(a)).

Definition 2. Upper envelope

Let $\psi \subseteq \mathcal{S}^{\ell_2}$ be an arbitrary subset. The *Upper envelope* of ψ is the union of line segments such that each line segment is either (i) part of ℓ_2 which is not covered by any square in ψ or (ii) part of boundary of the area $(\underset{s \in \psi}{\cup} s) \cap \mathcal{H}$ where every point on the boundary lies above ℓ_2 (see thick lines in Figure 4.5(b)).



(a) (b)

Figure 4.5: (a) Lower envelope, and (b) Upper envelope

Note that, in the definition of *triplet* $T = (u, v, x)$, x is the value equal to the maximum of (i) x-coordinate of the point at which the square u begins to appear on the *lower envelope* and (ii) x-coordinate of the point at which the square v begins to appear on the *upper envelope*.

In our algorithm for the SSC problem, we use the *triplets* $\mathcal{T}$ to represent the sweep-line *status* while performing a plane sweep. Hence, from now onward, we use the terms *triplet* and *status* interchangeably depending on the context. We build the cover of points sequentially at the time of sweeping the plane (containing strip $\mathcal{H}$) from left to right. Let $T \in \mathcal{T}$ be an arbitrary *triplet*. We now define a *successor* of *status* $T = (u, v, x)$, as follows:

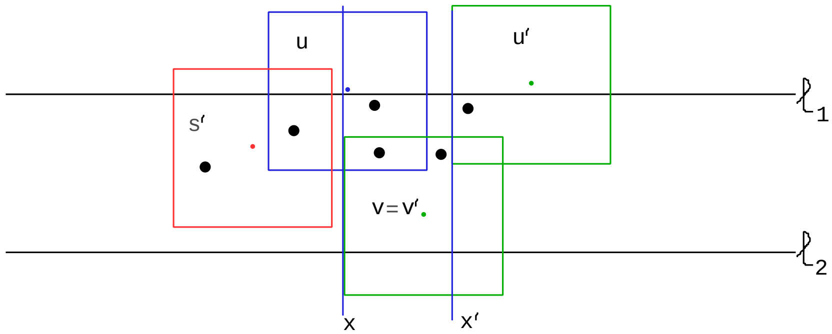
Definition 3. *Successor* **of** *status* $T = (u, v, x)$
Let $\mathcal{S}_u^{\ell_1} = \{s \in \mathcal{S}^{\ell_1} | x(s) > x(u)\}$ and $\mathcal{S}_v^{\ell_2} = \{s \in \mathcal{S}^{\ell_2} | x(s) > x(v)\}$. Consider a *status* $T' = (u', v', x')$ such that

(a) either (i) $u' = u$ and $v' \in \mathcal{S}_v^{\ell_2}$ or (ii) $v' = v$ and $u' \in \mathcal{S}_u^{\ell_1}$, and

(b) $x' \geq x$.

Let $[\ell, \ell']$ be a vertical strip bounded by vertical lines ℓ and ℓ' at coordinates x and x' of T and T' respectively. If all the points in $\mathcal{P} \cap [\ell, \ell']$ are covered by $\{u\} \cup \{v\}$, then T' is said to be a *successor* of T (see Figure 4.6).

Let $T^l = (s_1^l, s_2^l, x^l)$ and $T^r = (s_1^r, s_2^r, x^r)$ denote the *triplets* corresponding to dummy squares in $\mathcal{S}$. We now construct a directed graph $G = (V, E)$ by having a node $\nu \in V$ for



(a)

Figure 4.6: *Successor* $T' = (u', v', x')$ of *Triplet* $T = (u, v, x)$

each *triplet* $T \in \mathcal{T}$ and a directed edge $e = (\nu, \nu') \in E$ if and only if the corresponding *status* T' of node ν' is a *successor* of the corresponding *status* T of node ν. Let the node $\nu_l \in V$ correspond to *status* T^l and node $\nu_r \in V$ correspond to *status* T^r. Notice that the node ν_l has in-degree 0 and the node ν_r has out-degree 0. From the definition of *triplet* $T \in \mathcal{T}$, the total number of *triplets* $|\mathcal{T}| = O(m^3)$. Therefore, $|V| = |\mathcal{T}| = O(m^3)$. From the definition of *successor* of *status* T, the total number of *successors* of a *status* T is $O(m)$. Therefore, $|E| = O(m^4)$. Given the *status* T and the *status* T', verifying that T' is indeed a *successor* of T takes $O(n)$ time. Hence, the total time required to construct the graph G is $O(m^4 n)$.

Lemma 4.4.1. *If OPT is an optimal solution for the SSC problem, then there is a path from node ν_l to node ν_r, of length $|OPT| + 2$, in the directed graph G.*

Proof. We first set $OPT' = OPT \cup \{s_1^r, s_2^r\}$. Let x_j be the x-coordinate of the point at which the square $s_j \in OPT'$ begins to appear either on the *lower envelope* of $OPT' \cap \mathcal{S}^{\ell_1}$ or on the *upper envelope* of $OPT' \cap \mathcal{S}^{\ell_2}$. Without loss of generality let $OPT' = \{s_1, s_2, \ldots, s_{|OPT|+2} | x_1 \leq x_2 \leq \ldots \leq x_{|OPT|+2}\}$. We now set a *triplet* $T_0 = (u_0, v_0, x_0)$ where $u_0 = s_1^l$, $v_0 = s_2^l$ and $x_0 = \max(x(u_0) - \frac{1}{2}, x(v_0) - \frac{1}{2})$ and associate a *triplet* $T_j = (u_j, v_j, x_j)$ with each x_j for $j = 1, 2, \ldots, |OPT| + 2$ such that the *successor* of $T_j = (u_j, v_j, x_j)$ is $T_{j+1} = (u_{j+1}, v_{j+1}, x_{j+1})$ where either (i) $u_{j+1} = u_j$ and $v_{j+1} = s_{j+1}$ or (ii) $v_{j+1} = v_j$ and $u_{j+1} = s_{j+1}$, and $x_{j+1} \geq x_j$ for $j = 0, 1, \ldots, |OPT| + 2$. Now, starting with the *triplet* T_0 (=T^l) as the initial *status* of sweep line, we can sweep through OPT such that the sweep line changes its current *status* $T_j = (u_j, v_j, x_j)$ to its *successor*

Algorithm 4.3 Exact_Strip_Square_Cover($\mathcal{P}, \mathcal{S}, \mathcal{H}$)

```
Input: A set P of n points, a set S of m unit squares and a horizontal strip H of height 1.
Output: A subset OPT ⊆ S such that P ⊆ ∪_{s∈OPT} s.
Set S ← S ∪ {s_1^l, s_2^l, s_1^r, s_2^r}
Construct the set of triplets T from the set S.
Construct the directed graph G = (V, E) from the set of triplets T.
Compute a shortest path τ from ν_l to ν_r in G using breadth-first-search traversal.
Let OPT contain all squares from the triplets of τ
Reset OPT ← OPT \ {s_1^l, s_2^l, s_1^r, s_2^r}.
Return OPT
```

$T_{j+1} = (u_{j+1}, v_{j+1}, x_{j+1})$ until it reaches *status* T^r $(=T_{|OPT|+2})$. Note that, in this plane sweep process, the number of hops the sweep line makes from initial *status* to its final *status* is $|OPT| + 2$. Therefore, the corresponding nodes $\nu_0 = \nu_l$, ν_1, ν_2, …, $\nu_{|OPT|+1}$, $\nu_{|OPT|+2} = \nu_r$ constitute a path of length $|OPT| + 2$ in the graph G. Thus the lemma follows. □

From Lemma 4.4.1, it is known that there must be a path $\tau = \nu_l \sim> \nu_r$ of length $|OPT|+2$ between nodes ν_l and ν_r. To compute an optimal solution for a given instance of the SSC problem, we first construct the directed graph G and then find a shortest path between nodes ν_l and ν_r using breadth-first search. The sequence of steps required for computing an optimal solution OPT is given in Algorithm 4.3.

Lemma 4.4.2. *Algorithm 4.3 computes an optimum solution in* $O(m^4 n + n \log n)$ *time for the SSC problem.*

Proof. From Lemma 4.4.1, there is a path of length $|OPT| + 2$ between ν_l and ν_r in the directed graph G. Algorithm 4.3 computes a shortest path τ using the breadth-first-search technique. The length of τ is $|OPT| + 2$. Hence, the set of squares in the corresponding *successors* of this shortest path must be the same with the optimal solution. The running time of the algorithm is dominated by the time required to construct directed graph G plus the time required to arrange points in $\mathcal{P}$. Thus, the lemma follows. □

Algorithm 4.4 Discrete_Unit_Square_Cover($\mathcal{P}, \mathcal{S}$)

Input: A set $\mathcal{P}$ of n points, a set $\mathcal{S}$ of m unit squares in 2D, and an integer $k(>2)$.

Output: A subset $\mathcal{S}_{out} \subseteq \mathcal{S}$ such that $\mathcal{P} \subset \bigcup_{s \in \mathcal{S}_{out}} s$.

Set $\mathcal{S}_{out} \leftarrow \emptyset$

Let $\mathcal{R}$ be the smallest axis-aligned rectangular region containing all points in $\mathcal{P}$ and the centers of all squares in $\mathcal{S}$. Partition $\mathcal{R}$ into horizontal strips $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_r$ of height 1 such that $\bigcup_{i=1}^{r} \mathcal{H}_i \supseteq \mathcal{R}$.

for $(i = 1, 2 \ldots, r)$ **do**

  $\mathcal{S}_i = \{s \in \mathcal{S} | \alpha(s) \in \mathcal{H}_i$, where $\alpha(s)$ is the center of square $s\}$

end for

Set $\mathcal{S}_0 \leftarrow \emptyset$, $\mathcal{S}_{r+1} \leftarrow \emptyset$

for $(i = 1, 2 \ldots, r)$ **do**

  $\mathcal{P}' = \mathcal{P} \cap \mathcal{H}_i$

  $\mathcal{S}' = \mathcal{S}_{i-1} \cup \mathcal{S}_i \cup \mathcal{S}_{i+1}$

  $\mathcal{S}_{out} = \mathcal{S}_{out} \cup$ Exact_Strip_Square_Cover($\mathcal{P}', \mathcal{S}', \mathcal{H}_i$) /* Call Algorithm 4.3 */

end for

Return $\mathcal{S}_{out}$

Theorem 4.4.3. *Algorithm 4.4 is a* 2*-approximation algorithm in* $O(m^4 n + n \log n)$ *time for the DUSC problem.*

Proof. Follows from the same argument as in Theorem 4.3.1, and Lemma 4.4.2 (instead of Theorem 4.2.4). □

4.5 Conclusion

In this chapter we have proposed $(2 + \frac{4}{k-2})$-approximation algorithm for the DUSC problem, where $k(>2)$ is an integer. The time complexity of our proposed approximation algorithm is $O(km^k n)$, which is faster than the best known algorithm available in the literature for $k \in \{5, 6, \ldots, 8\}$ by sacrificing some approximation factor [78]. We also have proposed a 2-approximation algorithm for the DUSC problem using a technique similar to kind of [78]. The running time of the proposed algorithm for DUSC problem is $O(m^4 n + n \log n)$, which is an improvement by a factor of $O(m^4 n)$ over the best known algorithm available in the literature [78, 62].

Chapter 5

Constrained k-Center Problem on a Convex Polygon

In this chapter we consider the constrained k-center problem as follows:

> Constrained Convex Polygon Cover (CCPC): Given a convex polygon P with n vertices and an integer k, the objective is to cover the entire region of P using k congruent disks of minimum radius r_{opt}, centered on the boundary of P.

The CCPC problem has not been so well studied in the literature. Das et al. [24] studied the problem and developed a $(1 + \epsilon)$-approximation algorithm. Their algorithm covers the convex polygon P with k disks of radius $r \leq (1+\epsilon)r_{opt}$, but the centers of the k disks lie on only a specified edge of the polygon. Du and Xu [25] presented an approximation algorithm, which allows the centers to lie anywhere on the boundary of the polygon P. They first compute a rectangular W covering the convex polygon P, then cover W with k disks of smallest radius, centered on the boundary of W. They then move each of covering disks of W properly so that the centers of k disks lie on the boundary of P and the union of k disks covers P. In this chapter, we develop an algorithm, which uses a simple trick of carefully centering the disks on the boundary of P itself. Our algorithm achieves a better approximation to the CCPC problem for large values of k, when compared to the previous algorithms.

We first consider the decision version of the CCPC problem, which is useful to approximate the solution for the CCPC problem. We discuss an algorithm to solve the decision version of the CCPC problem approximately. For the decision version of the CCPC problem we propose an $(1 + \frac{7}{k})$-approximation algorithm, where $k \geq 7$. The running time of the algorithm is $O(n(n + k))$ time. For the CCPC problem, using the proposed algorithm for the decision version of the CCPC problem, we propose an $(1 + \frac{7}{k} + \frac{7\epsilon}{k} + \epsilon)$-approximation algorithm, which runs in $O(n(n+k)(|\log r_{opt}| + \log\lceil\frac{1}{\epsilon}\rceil))$ time for any $\epsilon > 0$, where n is the number of vertices of polygon P. The best known approximation factor of the algorithm in the literature for the CCPC problem is 1.8841 [25]. The running time of the 1.8841-approximation algorithm is $O(nk)$.

In Section 5.1, we present an algorithm for the decision version of the CCPC problem. In Section 5.2, we present an approximation algorithm for the CCPC problem. Finally, we conclude the chapter in Section 5.3.

5.1 Decision Version of CCPC Problem

In this section we present an algorithm to solve the decision version of the CCPC problem approximately. The decision version of the problem is as follows:

> k-COVER(P, k, r): Given a convex polygon P, an integer k and a real number r, check whether P has a cover with k congruent disks of radius r centered on ∂P, where ∂P is the boundary of polygon P.

Let $dist(p', p'')$ denote the Euclidean distance between two points p' and p''. For any two points p and q, $\overline{pq}$ denotes the line segment joining p and q. For any disk d_i, let ∂d_i be its boundary arc and the center of d_i be (x_i, y_i). Let the convex polygon P be placed such that it is lying to the right of y-axis (see Figure 5.1).

5.1.1 Preprocess

Let $P = (v_1, v_2, \ldots, v_n)$ be a convex polygon. Here, we first perform alignment on P as follows: For each vertex v_j, $1 \leq j \leq n$, we identify a vertex $v_{j'}$ such that $dist(v_j, v_{j'}) \geq dist(v_j, v_{j''})$ for all j'', $1 \leq j'' \leq n$ (in the case of more than one vertex, pick an arbitrary one). For each such pair of vertices $(v_j, v_{j'})$, we align P as follows: v_j and $v_{j'}$ are lying on x-axis and v_j is in right side of $v_{j'}$. For each such alignment, we apply the

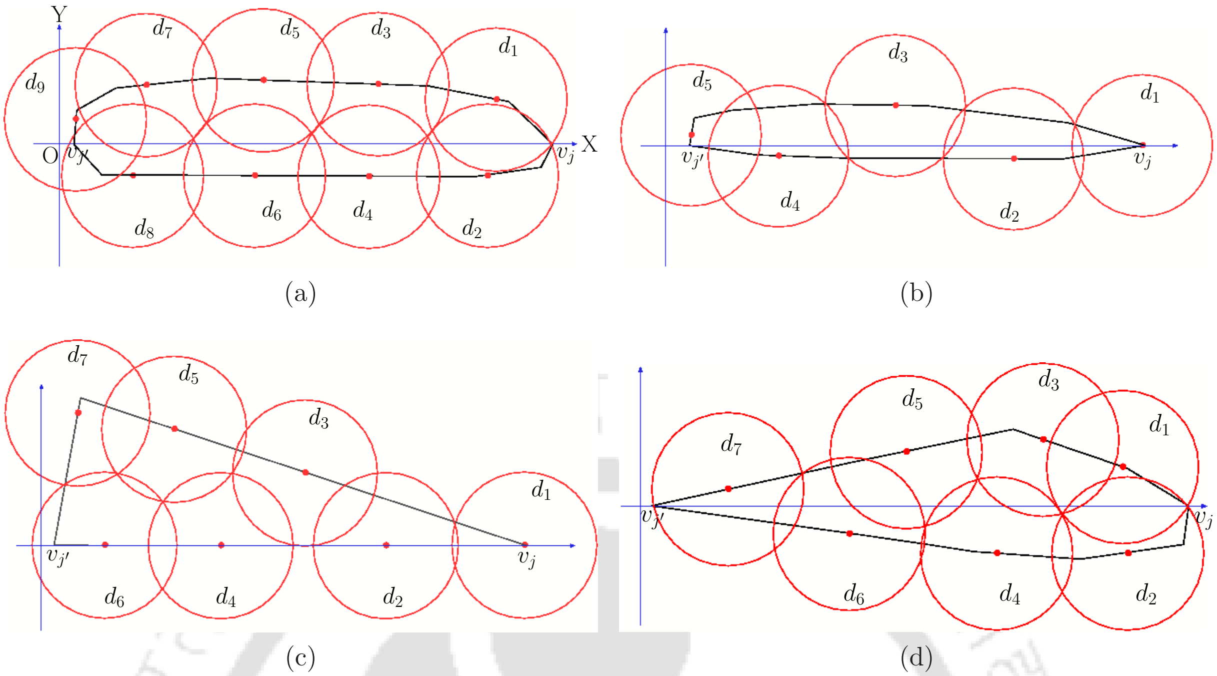


Figure 5.1: Constrained placement of disks

covering algorithm (Algorithm 5.1) to cover P by placing disks d_i ($i \geq 1$) on ∂P one by one from right to left, where the initial disk d_1 is either centered at v_j (see Figure 5.1(b)) or centered on ∂P (in counter-clockwise direction of v_j) so that ∂d_1 passes through v_j (see Figure 5.1(a)).

5.1.2 Approximation Algorithm for the Decision Version of the Problem

Definition 5.1.1. *Upper chain* (resp. *Lower chain*) of P is the locus of points on ∂P starting from the right most vertex v_j of P to the left most vertex $v_{j'}$ in counter-clock (resp. clock) wise order. We use the notations U_c and L_c to denote the *upper chain* and *lower chain* respectively.

Definition 5.1.2. *Disk-constrained placement* of disks is a placement of disks d_i ($i \geq 3$) on the boundary of P from right to left, centered alternately on U_c and L_c of P such that the current disk d_i is centered at leftmost point on U_c or L_c satisfying (i) ∂d_i passes through the leftmost intersection point between the disks d_{i-1} and d_{i-2} for $i \geq 3$, and

(ii) d_2 contains the vertex v_j or d_i contains the leftmost intersection point between d_{i-2} and L_c or d_{i-2} and U_c for $i \geq 3$ (see Figure 5.1(a)).

Definition 5.1.3. *Chain-constrained placement* of disks is the placement of disks d_i ($i \geq 2$) on the boundary of P from right to left, centered alternately on L_c and U_c of P such that the current disk d_i is centered at the leftmost point on L_c or U_c satisfying the following condition: d_i contains the leftmost intersection point between (i) d_{i-1} and L_c, and (ii) d_{i-1} and U_c (see Figure 5.1(b)).

A high level description of our algorithm for the decision version of the CCPC problem is as follows: we first find the proper alignment of P as described in subsection 5.1.1. Next we place congruent disks of radius r centered on ∂P by using the greedy strategies in Definition 5.1.2 and Definition 5.1.3 (*disk-* and/or *chain-constrained placement*) until P is covered by the union of these disks provided P is coverable by disks of radius r. Let ℓ be the number of congruent disks of radius r centered on ∂P such that the union of ℓ disks covers P. Next, we increase the radius of the first k disks, to $r' = r + \frac{(\ell-k)}{k} r$ and reposition their centers on ∂P such that every disk satisfies the *constrained placement* requirement. Finally, we remove the disks placed after k-th disk. The detailed pseudo-code of our strategy is described in Algorithm 5.1, 5.2, 5.3, and 5.4. The outline of our algorithm for the decision version of the CCPC problem is summarized by the following steps.

(a) Align the convex polygon P (as described in subsection 5.1.1).

(b) Run Algorithm 5.1 to place ℓ congruent disks of given radius r, on ∂P using *disk-* and/or *chain-constrained placement* until P is covered by the union of these ℓ disks provided P is coverable by disks of radius r.

(c) Run Algorithm 5.4 to reset the radius of the first k disks to $r' = r + \frac{(\ell-k)}{k} r$, reposition their centers on ∂P such that every disk satisfies *disk-* and/or *chain-constrained placement* requirement and finally, remove $(\ell - k)$ redundant disks.

In Algorithm 5.2, if switching happens from *disk-constrained placement* to *chain-constrained placement*, then the current disk d_i covers the left intersection point between d_{i-1} and d_{i-2} (it must be within P otherwise it would not be switching from

Algorithm 5.1 ℓ-COVER(P, r)

Input: Aligned convex polygon P, a real number r.
Output: Set $\mathcal{D} = \{d_1, d_2, \ldots, d_\ell\}$ of disks of radius r, centered on ∂P such that $P \cap (\bigcup_{d \in \mathcal{D}} d) = P$ provided P is coverable by k congruent disks of radius r.
Place the disk d_1 centered at the right-most vertex v_j after alignment of convex polygon P.
if d_2 cannot be placed on the lower chain by *chain-constrained placement* **then**
    Reposition d_1 such that ∂d_1 passes through v_j and d_1 is centered on the upper chain.
    Place d_2 on the lower chain such that ∂d_2 passes through v_j.
    Set $\mathcal{D} \leftarrow \{d_1, d_2\}$
else
    Place d_2 on the lower chain by *chain-constrained placement*
    Set $\mathcal{D} \leftarrow \{d_1, d_2\}$
end if
$\mathcal{D} = constrained\_placement(P, \mathcal{D}, r)$ //Call Algorithm 5.2
if $(P \cap (\bigcup_{d \in \mathcal{D}} d) = P)$ **then**
    $\ell \leftarrow |\mathcal{D}|$
    Return $(\mathcal{D}, \ell)$
else
    Return $(\emptyset, 0)$
end if

disk-constrained placement to *chain-constrained placement*) in addition it covers left intersection points between U_c and the last disk (d_{i-1} or d_{i-2}) placed on U_c, and between L_c and the last disk (d_{i-1} or d_{i-2}) placed on L_c respectively (see Figure 5.1(d)). On the other hand, if switching happens from *chain-constrained placement* to *disk-constrained placement*, then ∂d_i passes through the left intersection point (i) between U_c and the disk d_{i-1} if d_{i-1} is centered on L_c or (ii) between L_c and the disk d_{i-1} if d_{i-1} is centered on U_c (see Figure 5.1(c)).

Whenever Algorithm 5.2 places a disk d_i, Algorithm 5.3 is invoked. If d_i is placed by *disk-constrained placement*, Algorithm 5.3 checks whether the remaining uncovered portion of P can be covered, by exhaustively placing at most 3 disks. If this portion of P is covered, Algorithm 5.3 returns these disks to Algorithm 5.2.

Notations : Let $\mathcal{D}$=$\{d_1, d_2, \ldots, d_\ell\}$ be the set of disks placed by Algorithm 5.1 and $\mathcal{D} \backslash \{d_{k+1}, d_{k+2}, \ldots, d_\ell\}$ be the set of disks retained to be centered on the boundary after

Algorithm 5.2 constrained_placement(P, $\mathcal{D}$, r)

Input: Aligned convex polygon P, set $\mathcal{D}$ of disks, and radius r.
Output: Set $\mathcal{D} = \{d_1, d_2, \ldots\}$ of disks of radius r, centered on ∂P such that $P \cap (\bigcup_{d \in \mathcal{D}} d) = P$ provided P is coverable by k congruent disks of radius r.
$i \leftarrow 3$
while $((P \cap (\bigcup_{d \in \mathcal{D}} d) \neq P)$) **do**
    $\mathcal{D}^1 = non\_constrained\_placement(P, \mathcal{D}, r)$ //Call Algorithm 5.3
    if $(P \cap (\bigcup_{d \in \mathcal{D}^1} d) = P)$ **then**
        Set $\mathcal{D} \leftarrow \mathcal{D}^1$
        Return($\mathcal{D}$)
    end if
    if d_{i-1} is centered on the lower chain **then**
        if d_i cannot be placed on the upper chain by *chain-constrained placement* **then**
            if d_i cannot be placed on the upper chain by *disk-constrained placement* **then**
                Return ($\emptyset$)
            end if
            Place d_i on the upper chain by *disk-constrained placement* (see Definition 5.1.2)
        else
            Place d_i on the upper chain by *chain-constrained placement* (see Definition 5.1.3)
        end if
    else
        if d_i cannot be placed on the lower chain by *chain-constrained placement* **then**
            if d_i cannot be placed on the lower chain by *disk-constrained placement* **then**
                Return ($\emptyset$)
            end if
            Place d_i on the lower chain by *disk-constrained placement*
        else
            Place d_i on the lower chain by *chain-constrained placement.*
        end if
    end if
    $\mathcal{D} = \mathcal{D} \cup \{d_i\}$, $i \leftarrow i + 1$
end while
Return($\mathcal{D}$)

increasing the radius and removing the redundant disks in Algorithm 5.4. Let $\mathcal{D}' = \{d'_1, d'_2, \ldots, d'_k\}$ be the set of disks in an optimal solution of k-COVER(P, k, r) and (x'_i, y'_i) be the center of the disk d'_i ($i = 1, 2, \ldots, k$). Let $\alpha(i)$ denote the disk $d'_s (\in \mathcal{D}')$ such that $x_{i+2} \leq x'_s \leq x_i$ and centered on the same chain as d_i and d_{i+2} or $x_{i+3} \leq x'_s \leq x_{i+1}$

Algorithm 5.3 non_constrained_placement(P, $\mathcal{D}$, r)

Input: Aligned convex polygon P, set $\mathcal{D}$ of disks and radius r.
Output: Set $\mathcal{D}^1 = \{d_1, d_2, \ldots\}$ of disks of radius r, centered on ∂P such that $P \cap (\bigcup_{d \in \mathcal{D}^1} d) = P$ if uncovered region in P can be covered with at most 3 disks along with $\mathcal{D}$.
$i \leftarrow |\mathcal{D}| + 1$, set $\mathcal{D}^1 \leftarrow \mathcal{D}$
if d_{i-1} is centered by *disk-constrained placement* **then**
  Place a disk d of radius r centered at the left intersection point between d_{i-1} and d_{i-2}. /* left intersection point is inside P, not on U_c or L_c because d_{i-1} is placed by *disk-constrained* placement */ (see Figure 5.2)
  Let t_1, t_2, …, t_m be the intersection points of ∂P with the disk d in order on ∂P starting from center of d_{i-1} (t_1) to center of d_{i-2} (t_m). (see Figure 5.2)
  for ($s = 2, 3, \ldots, m-1$) **do**
    Place the disk d_i centered at t_s and set $\mathcal{D}^1 = \mathcal{D}^1 \cup \{d_i\}$
    if ($P \cap (\bigcup_{d \in \mathcal{D}^1} d) = P$) **then**
      Return ($\mathcal{D}^1$)
    else
      if there are two uncovered components in ($P - (P \cap (\bigcup_{d \in \mathcal{D}^1} d))$) **then**
        Let Δ_1 and Δ_2 be the uncovered components of ($P - (P \cap (\bigcup_{d \in \mathcal{D}^1} d))$) lying below and above d_i respectively
        Place d_{i+1} and d_{i+2} centered on ∂P such that ($\Delta_1 \cap d_{i+1} = \Delta_1$) and ($\Delta_2 \cap d_{i+2} = \Delta_2$) (see Lemma 5.1.6)
        $\mathcal{D}^1 = \mathcal{D}^1 \cup \{d_{i+1}, d_{i+2}\}$
        Return ($\mathcal{D}^1$)
      end if
    end if
  end for
end if
Return($\mathcal{D}$)

and centered on the same chain as d_{i+1} and d_{i+3}, where $1 \leq s \leq k$.

Lemma 5.1.4. *If the disks d_1, d_2, …, d_ℓ are centered on ∂P by constrained placement (while-loop in line 4 of Algorithm 5.2), then $(\bigcup_{i=1}^{\ell} d_i) \cap P = P$.*

Proof. In every iteration of the while-loop in line 4 of Algorithm 5.2, the disk d_i is placed on either U_c or L_c such that its ∂d_i passes through the left intersection point between d_{i-1} and d_{i-2} (see Figure 5.1). Also d_i covers the nearest intersection point between ∂P and ∂d_{i-2} (if d_i, d_{i-1} and d_{i-2} are placed by *disk-constrained placement*) or d_i covers both

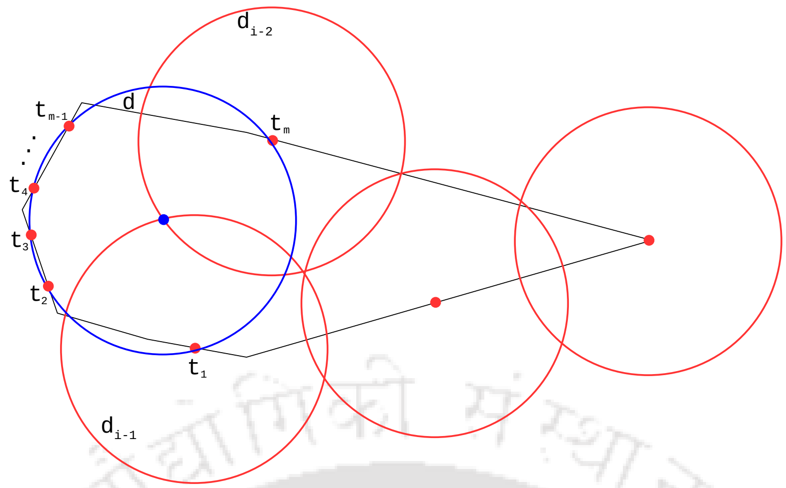


Figure 5.2: Non-constrained placement

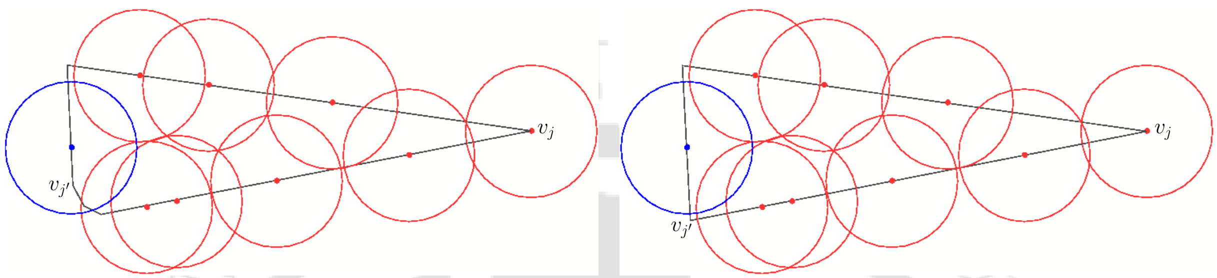
the left intersection points between (i) previously placed disk and U_c, and (ii) previously placed disk and L_c (if d_i and d_{i-1} are placed by *chain-constrained placement*). Thus, the lemma follows. □

Lemma 5.1.5. *If the disks d_1, d_2, $\ldots$, d_ℓ are all placed only by chain-constrained placement in Algorithm 5.2, then $\ell \leq k+1$.*

Proof. The proof follows from the fact that the center of at least one optimal disk must lie within disk d_i, for every $i \geq 2$, placed by *chain-constrained placement* in Algorithm 5.2 since the x-coordinate of the center of d_i is the smallest (see Figure 5.1(b)). □

Lemma 5.1.6. *At most three disks are required to be placed by non-constrained placement in Algorithm 5.3 to cover P.*

Proof. In line 5 of Algorithm 5.2, Algorithm 5.3 is invoked to test whether the remaining uncovered portion of P can be covered with at most three disks (i.e, we have reached the end of P). If the left intersection point of d_{i-1} and d_{i-2} needs to be covered by disk d_i using *non-constrained placement* (see Figure 5.3), then let Δ_1 and Δ_2 be the uncovered regions above and below d_i respectively. Without loss of generality let d_{i-1} be centered on L_c and d_{i-2} on U_c. Let $p_i = (x_i, y_i)$, $p_{i-1} = (x_{i-1}, y_{i-1})$ and $p_{i-2} = (x_{i-2}, y_{i-2})$ where (x_f, y_f) is the center of disk d_f. Then observe that $dist(p_i, p_{i-1}) \leq 2r$ and $dist(p_i, p_{i-3}) \leq 2r$. The following two statements must be true because otherwise it will contradict that the polygon P is a convex polygon: (i) A disk d of radius r centered at the mid-point on line segment $\overline{p_i, p_{i-1}}$ must contain p_i, p_{i-1} and Δ_2, and (ii) a disk $\tilde{d}$ of radius r centered at the mid-point on line segment $\overline{p_i, p_{i-2}}$ must contain p_i, p_{i-2} and Δ_1. Therefore, the uncovered regions Δ_1 and Δ_2 (if they exist) above and below d_i must be covered by at most one disk each, because of the convexity of P. □



(a) (b)

Figure 5.3: Blue colored disk centered by non-constrained placement

Algorithm 5.2 places as many as optimal number of disks plus one by *chain-constrained placement* only (see Lemma 5.1.5). Therefore, from now onward we restrict our discussion to *disk-constrained placement*. For any consecutively placed disks d_i and d_{i+1} (without loss of generality, centered on L_c and U_c of P respectively) by *disk-constrained placement* in Algorithm 5.2, let p_1 and p_2 be the intersection points between ∂d_i and ∂d_{i+1} such that $x_{p_1} \leq x_{p_2}$ (for $i \geq 3$), where x_{p_1} and x_{p_2} are the x-coordinates of p_1 and p_2 respectively (see Figure 5.4(a)). Let d'_j and $d'_{j'}$ be two left-most disks centered on L_c and U_c respectively such that $x_i < x'_j$ and $x_{i+1} < x'_{j'}$ (Note that x_i, x_{i+1}, x'_j, $x'_{j'}$ are x-coordinates of the centers of d_i, d_{i+1}, d'_j, $d'_{j'}$ respectively). Let l_{p_1} be a vertical line passing through p_1. Let p'_1 and p'_2 be the left and right intersection points between $\partial d'_j$ and $\partial d'_{j'}$ such that $x_{p'_1} \leq x_{p'_2}$, where $x_{p'_1}$ and $x_{p'_2}$ are the x-coordinates of p'_1 and p'_2, respectively.

Observation 5.1.7. For any two consecutively placed disks d_i and d_{i+1} ($i \geq 3$), p'_1 cannot lie to the left of l_{p_1}.

Proof. Assume that p'_1 is lying to the left of l_{p_1}. Since p'_1 is lying to the left of l_{p_1} and d'_j, $d'_{j'}$ are two left-most disks (centered on L_c and U_c respectively) in the optimum solution such that $x_i < x'_j$ and $x_{i+1} < x'_{j'}$, both d'_j and $d'_{j'}$ intersect l_{p_1} and at least one of them contain the point p_1. Without loss of generality let the disk containing p_1 be $d'_{j'}$ centered on U_c. Let q_1 and q_2 be the intersection points between ∂d_{i-1} and ∂d_{i-2} such that $x_{q_1} \leq x_{q_2}$. By *disk-constrained placement* of disks, q_1 is lying to the right of l_{p_1}. Let the centers of disks d_{i+1}, $d'_{j'}$ and d_{i-1} be labeled as x, y and z respectively. Let the horizontal lines through x, y and z intersect l_{p_1} at a, b and c respectively (see Figure 5.4(b)). Let the length of line segments $dist(p_1, x) = r_1 = r$, $dist(p_1, y) = r_2$, $dist(p_1, z) = r_3$. Note that (i) $r_3 > r_1$ due to *disk-constrained placement* of disks and (ii) $r_1 > r_2$ (by assumption). Therefore, $r_3 > r_1 > r_2$ and $\theta_1 < \theta_2 < \theta_3$, where $\theta_1 = \angle ap_1x$, $\theta_2 = \angle bp_1y$, $\theta_3 = \angle cp_1z$. Therefore there exists a reflex vertex between points x and z on U_c, contradicting that P is convex. □

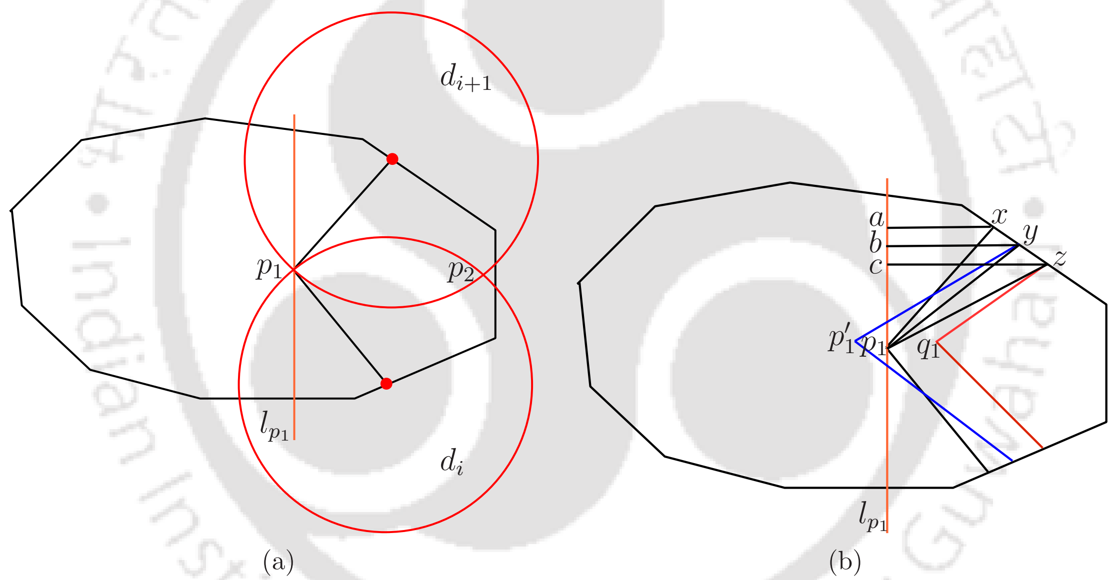


Figure 5.4: Proof of Observation 5.1.7

In the lemmata 5.1.8, 5.1.9 and 5.1.10, we prove that there must be at least one optimal disk in every subsequence of four consecutive disks placed by *disk-constrained placement*, which leads to the result: $\ell \leq k + 1$ (i.e. at most $k + 5$ disks are placed by Algorithm 5.1 to cover P), where k is the optimal number of disks for a given radius r.

Lemma 5.1.8. *For any consecutively placed disks $d_i, d_{i+1}, d_{i+2}, d_{i+3}$, $(i \geq 3)$, there exists at least one disk d'_q centered at (x'_q, y'_q) such that $\alpha(i) = d'_q$ i.e., $x_{i+2} \leq x'_q \leq x_i$ or $x_{i+3} \leq x'_q \leq x_{i+1}$, $1 \leq q \leq k$, where (x_f, y_f) denote the center of disk d_f.*

Proof. Let $p_1 = (x_{p_1}, y_{p_1})$ and $p_2 = (x_{p_2}, y_{p_2})$ be the intersection points between ∂d_i and ∂d_{i+1} such that $x_{p_1} \leq x_{p_2}$. Let l_{p_1} be a vertical line passing through p_1 (see Figure 5.5). On the contrary of the lemma, assume that no disk d'_q from the optimal solution is centered on ∂P such that d'_q is centered on L_c and $x_{i+2} \leq x'_q \leq x_i$ or d'_q is centered on U_c and $x_{i+3} \leq x'_q \leq x_{i+1}$, $1 \leq q \leq k$ (see Figure 5.5). Let d'_j and $d'_{j'}$ be two disks from the optimal solution centered left most but to the right of (x_i, y_i) and (x_{i+1}, y_{i+1}) respectively, that is, $x_i < x'_j$ and $x_{i+1} < x'_{j'}$. Note that (x_f, y_f) is the center of the disk d_f. Let p'_1 and p'_2 be the intersection points between $\partial d'_j$ and $\partial d'_{j'}$. By observation 5.1.7, p'_1 is lying to the right of vertical line l_{p_1}. If $dist(p'_1, (x_{i+2}, y_{i+2})) \geq r$, then there must be a disk d'_q centered on ∂P such that $x_{i+2} \leq x'_q$, otherwise the disk d'_q cannot cover p'_1. If $dist(p'_1, (x_{i+2}, y_{i+2})) < r$, some disk d'_q has to be centered such that $x_{i+2} > x'_q$. This implies that the disk d'_q must not cover p_1, otherwise it would contradict that the disk d_{i+2} is centered at (x_{i+2}, y_{i+2}) as d_{i+2} could be moved with its new center to be closer to the y-axis than (x_{i+2}, y_{i+2}) while ∂d_{i+2} is still passing through p_1. Therefore, the distance between (x_{i+3}, y_{i+3}) and the left intersection point between $\partial d'_q$ and $\partial d'_{j'}$ is greater than r, implying that some other disk $d'_{q'}$ must be centered such that $x_{i+3} \leq x'_{q'} \leq x_{i+1}$. Thus the lemma follows ☐

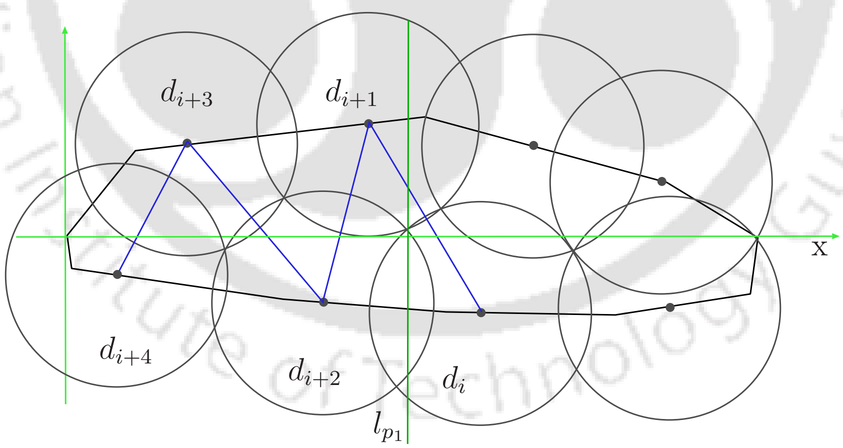


Figure 5.5: Proof of Lemmata 5.1.8 and 5.1.9

Lemma 5.1.9. *If* $\alpha(i) = d'_s$ *and* $\alpha(i+1) = d'_t$*, then* $s \neq t$*, where* $1 \leq s, t \leq k$.

Proof. Let the disks d_i, d_{i+2}, d_{i+4} and d_{i+1}, d_{i+3} be centered on the L_c and the U_c of P, respectively (see Figure 5.5). By Lemma 5.1.8, there exists at least one disk d'_s centered on L_c between the centers of d_i and d_{i+2} or centered on U_c between the centers of d_{i+1} and d_{i+3}. Then, we consider the following two cases:

(a) If the disk d'_s is placed on L_c between d_i and d_{i+2}, then by the same argument as in Lemma 5.1.8, for consecutively placed disks d_{i+1}, d_{i+2}, d_{i+3} and d_{i+4}, there must be at least one disk d'_t such that $\alpha(i+1) = d'_t$, where $1 \leq s, t \leq k$ and $s \neq t$.

(b) If the disk d'_s is placed on U_c between d_{i+1} and d_{i+3}, then the center of the disk d'_t placed on L_c, covering the left intersection point between the disks d'_j and $d'_{j'}(= d'_s)$ (to the right of centers of d_i and d_{i+3}, respectively) cannot lie to the left of the center of d_{i+4} on L_c since d_{i+4} does not intersect with d_i and d_{i+1} (otherwise there is no need for d_{i+2}). □

Lemma 5.1.10. *If k disks of radius r are sufficient to cover P entirely, then at most $k+7$ disks of radius r are required to cover P by Algorithm 5.1.*

Proof. Case (i) if no switching occurs in Algorithm 5.2 and all ℓ disks are placed by *disk-constrained placement*: Let $\mathcal{D}_i$ be the set of disks in the optimal solution $\mathcal{D}'$ corresponding to disks d_i, d_{i+1}, d_{i+2} and d_{i+3} such that every $d \in \mathcal{D}_i(\subseteq \mathcal{D}')$ is centered either between the centers of d_i and d_{i+2} and on the same chain as d_i and d_{i+2} or between the centers of d_{i+1} and d_{i+3} and on the same chain as d_{i+1} and d_{i+3}. By Lemma 5.1.9, $|\mathcal{D}_i \cap \mathcal{D}_{i+1}| < \max(|\mathcal{D}_i|, |\mathcal{D}_{i+1}|)$ and $|\mathcal{D}_i| \geq 1$ for $i \geq 3$. Then

$$k = |\mathcal{D}'| \geq \left| \bigcup_{i=3}^{\ell-3} \mathcal{D}_i \right| \geq \sum_{i=3}^{\ell-3} 1 = \ell - 5 \implies \ell \leq k+5.$$

Case (ii) if switching occurs in Algorithm 5.2 and a sequence of ℓ disks placed by Algorithm 5.1 are using *chain-constrained*, *disk-constrained* and *chain-constrained placement*: Let ℓ_1, ℓ_2 and ℓ_3 be the subsequences of consecutively placed disks by *chain-constrained*, *disk-constrained* and then *chain-constrained placement* again respectively. Then $\ell = \ell_1 + \ell_2 + \ell_3$. By lemma 5.1.5 and case (i), $\ell \leq k+7$. □

Lemma 5.1.11. *The running time of Algorithm 5.1 is $O(n + \ell)$.*

Proof. Initially, Algorithm 5.1 places two disks to cover P and invokes Algorithm 5.2 to cover the remaining uncovered portion of P. Algorithm 5.2 invokes Algorithm 5.3 to check whether the remaining uncovered region of P can be covered with at most three disks using *non-constrained placement*. Algorithm 5.3 computes at most m candidate locations for the centers of these three disks (d_i, d_{i+1} and d_{i+2}) if d_{i-1} is centered by *disk-constrained placement*, where $m \leq 4$, using at least $(\ell - 4)$ iterations of the while-loop at line 4 of Algorithm 5.2, and in the last iteration $m = O(n)$. Hence, the running time of Algorithm 5.3 is $O(n)$. Algorithm 5.2 places disks one by one either using *chain-constrained placement* or *disk-constrained placement* until P is fully covered. Algorithm 5.2 places at most ℓ disks by keeping track of the uncovered portion of ∂P. Hence, the running time of Algorithm 5.2 is $O(n + \ell)$. The running time of Algorithm 5.1 follows from the running times of Algorithm 5.2 and Algorithm 5.3. Thus the lemma follows. ☐

Algorithm 5.4 k-COVER(P, k, r)

Input: Convex polygon P, a positive integer k and a real number r (radius).
Output: Set $\mathcal{D} = \{d_1, d_2, \ldots, d_k\}$ of disks of radius $r' \leq (1 + \frac{7}{k})r$, centered on ∂P such that $P \cap (\bigcup_{d \in \mathcal{D}} d) = P$ if P is coverable with k disks of radius r.
for ($j = 1, 2, \ldots, n$) **do**
  Compute the farthest vertex $v_{j'}$ from vertex v_j.
  Align P such that v_j and $v_{j'}$ are lying on x-axis, x-coordinate of v_j is greater than the x-coordinate of $v_{j'}$ and the whole P lies to the right of the y-axis.
  ($\mathcal{D}$, ℓ)= ℓ-COVER(P, r) //Run Algorithm 5.1
  if $((P \cap (\bigcup_{d \in \mathcal{D}} d)) = P)$ **then**
    $\mathcal{D} = \mathcal{D} \setminus \{d_{k+1}, d_{k+2}, \ldots, d_\ell\}$
    Reset the radius of every $d \in \mathcal{D}$ to $r' = (1 + \frac{(\ell - k)}{k})r$
    Drag the centers of every disk $d \in \mathcal{D}$ from right to left towards the y-axis such that every d satisfies the *constrained placement* requirement.
    Return ($\mathcal{D}$, r')
  end if
end for
Return ($\emptyset$, 0)

Lemma 5.1.12. *The running time of Algorithm 5.4 is* $O(n(n + k))$.

Proof. The for-loop in line 3 of Algorithm 5.4 runs (in the worst case) for every vertex of convex polygon P. In every iteration of the for-loop (line number 3 of Algorithm 5.4), we compute the farthest vertex $v_{j'}$ from a vertex v_j in $O(n)$ time (line number 4 of Algorithm 5.4). In line 6, we invoke Algorithm 5.1, which takes $O(n+\ell)$ time (see Lemma 5.1.11). Again lines 8-10 take $O(n+k)$ time. From Lemma 5.1.10 we know that $\ell \leq k+7$. Therefore, the running time of Algorithm 5.4 is $n(n+(n+k+7)+(n+k)) = O(n(n+k))$ time. □

Theorem 5.1.13. *Algorithm 5.4 is $(1+\frac{7}{k})$-approximation algorithm ($k \geq 7$) for the decision version of the CCPC problem.*

Proof. Let $\mathcal{D}$=$\{d_1, d_2, \ldots, d_\ell\}$ be the set of disks centered on ∂P to cover P by Algorithm 5.1. Now, the number of disks centered on L_c after the disk d_k is placed is at most $\lceil\frac{(\ell-k)}{2}\rceil$ and the number of disks centered along L_c starting from the vertex v_j of P to the center of d_k is at most $\lceil\frac{k}{2}\rceil$. If the radius of these $\lceil\frac{k}{2}\rceil$ disks centered on L_c is increased by ρ such that the area covered by $\lceil\frac{(\ell-k)}{2}\rceil$ disks centered on L_c after d_k, is covered by these $\lceil\frac{k}{2}\rceil$ disks, then $\rho = \frac{(\lceil\frac{(\ell-k)}{2}\rceil)r}{\lceil\frac{k}{2}\rceil} = \lceil\frac{(\ell-k)}{k}\rceil r$. Let the radius of every disk $d_i \in \mathcal{D}$, for $1 \leq i \leq k$, be increased by an additive factor ρ, where $\rho \leq \frac{7r}{k}$ because $(\ell-k) \leq 7$ (see Lemma 5.1.10). The centers of the disks d_1, d_2, d_3, $\ldots$, d_k on ∂P are moved left such that the *disk* and/or *chain-constrained placement* requirement is satisfied by every disk (see Figure 5.6). Therefore, the disks d_{k+1}, d_{k+2}, $\ldots$, d_ℓ will become redundant and can be removed. The radius $r' = r + \rho \leq (1+\frac{7}{k})r$. Thus the theorem follows. □

5.2 Constrained Convex Polygon Cover Problem

Here, we describe Algorithm 5.5 to solve the *constrained convex polygon cover* (CCPC) problem. Algorithm 5.5 covers P with at most k congruent disks of radius r' ($\leq (1+\delta)r_{opt}$), and centered on ∂P, where $\delta = \frac{7}{k} + \frac{7\epsilon}{k} + \epsilon$ and r_{opt} is the optimum radius of k congruent disks. To achieve this we first use the doubling technique as follows: if $r_{opt} > 1$, we invoke our algorithm (Algorithm 5.4) for the decision version of CCPC problem with radius equal to 2^j for every $j = 1, 2, \ldots, j^*$ until a cover of P is found for radius 2^{j^*}, where j^* is the smallest positive integer (lines 10-16 in Algorithm 5.5),

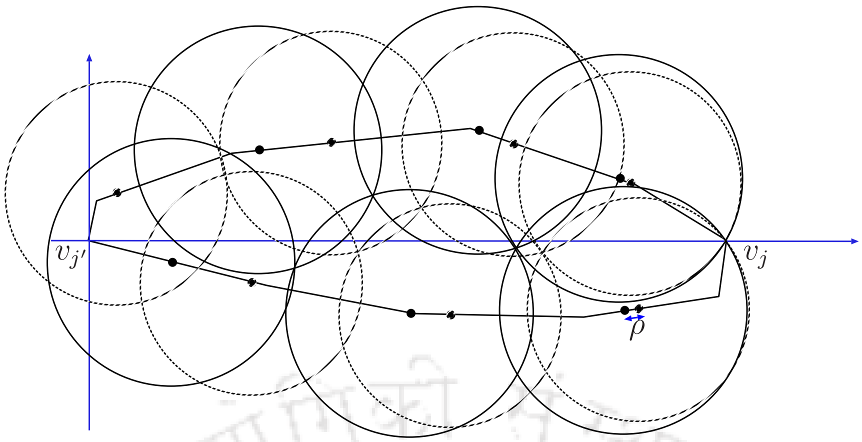


Figure 5.6: Proof of Theorem 5.1.13

otherwise we invoke Algorithm 5.4 with radius equal to 2^{-j} for every $j = 1, 2, \ldots, j^*$ till cover of P is found for radius 2^{-j^*}, where j^* is the largest positive integer (lines 4-9 in Algorithm 5.5). Therefore, r_{opt} belongs to either $[2^{j^*-1}, 2^{j^*}]$ or $[2^{-j^*-1}, 2^{-j^*}]$. Note that the size of this interval is at most r_{opt}. Let $[\mu, \nu]$ be the interval. Let $\gamma = \frac{\mu+\nu}{2}$. Now, we divide the interval $[\mu, \nu]$ into two intervals $[\mu, \gamma]$ and $[\gamma, \nu]$, and decide the interval that contains r_{opt}. Let this new interval be $[\mu, \nu]$ and repeat the same process $\log\lceil\frac{1}{\epsilon}\rceil$ times.

Lemma 5.2.1. *$(\nu - \mu) \leq \epsilon r_{opt}$, where μ, ν are the values after the for-loop in line 18 of Algorithm 5.5.*

Proof. Initially $\mu = 2^{j^*-1}$ and $\nu = 2^{j^*}$, where $2^{j^*-1} \leq r_{opt} \leq 2^{j^*}$. After the for-loop in line 18, the size of the interval is $(\nu - \mu) \leq \frac{(2^{j^*} - 2^{j^*-1})}{2^{\log\lceil\frac{1}{\epsilon}\rceil}} \leq \frac{r_{opt}}{2^{\log\lceil\frac{1}{\epsilon}\rceil}} \leq \epsilon r_{opt}$. The same proof follows for initial values $\mu = 2^{-j^*-1}$ and $\nu = 2^{-j^*}$. $\square$

Theorem 5.2.2. *Algorithm 5.5 is $(1+\delta)$-approximation algorithm for the CCPC problem, with running time $O(n(n+k)(|\log r_{opt}| + \log\lceil\frac{1}{\epsilon}\rceil))$, where $\delta = \frac{7}{k} + \frac{7\epsilon}{k} + \epsilon$, $k \geq 7$ and $\epsilon > 0$.*

Proof. The radius r_{opt} is initially made to lie in the interval $[2^{j^*-1}, 2^{j^*}]$ or $[2^{-j^*-1}, 2^{-j^*}]$ by the while-loop at line 6 or line 12 of Algorithm 5.5. Then, after the for-loop in line 18 of Algorithm 5.5, we reduce this interval to $[\mu, \nu]$ such that $\mu \leq r_{opt} \leq \nu$ and $(\nu - \mu) \leq \epsilon r_{opt}$ (Lemma 5.2.1). Therefore, $\nu \leq \mu + \epsilon r_{opt} \leq r_{opt} + \epsilon r_{opt} \leq (1 +$

Algorithm 5.5 COVER(P, k, ϵ)

1: **Input:** Convex polygon P, a positive integer k and an $\epsilon > 0$.
2: **Output:** Set $\mathcal{D} = \{ d_1, d_2, \ldots, d_k \}$ of k disks having equal radius $r' \leq (1 + \frac{7}{k} + \frac{7\epsilon}{k} + \epsilon) r_{opt}$, where $k \geq 7$ and an $\epsilon > 0$.
3: ($\mathcal{D}$, r')= k-COVER(P, k, 1) //Run Algorithm 5.4
4: **if** ($r' \neq 0$) **then**
5: $\quad j \leftarrow 1$
6: $\quad$ **while** ($r' \neq 0$) **do**
7: $\quad\quad j \leftarrow j - 1$
8: $\quad\quad$ ($\mathcal{D}$, r')= k-COVER(P, k, 2^{j-1}) //Run Algorithm 5.4
9: $\quad$ **end while**
10: **else**
11: $\quad j \leftarrow 0$
12: $\quad$ **while** ($r' = 0$) **do**
13: $\quad\quad j \leftarrow j + 1$
14: $\quad\quad$ ($\mathcal{D}$, r')= k-COVER(P, k, 2^j) //Run Algorithm 5.4
15: $\quad$ **end while**
16: **end if**
17: $\mu \leftarrow 2^{j-1}$, $\nu \leftarrow 2^j$
18: **for** ($i = 1, 2, \ldots, \lceil \log \lceil \frac{1}{\epsilon} \rceil \rceil$) **do**
19: $\quad \gamma \leftarrow \frac{\mu + \nu}{2}$
20: $\quad$ ($\mathcal{D}$, r')= k-COVER(P, k, γ) //Run Algorithm 5.4
21: $\quad$ **if** ($r' \neq 0$) **then** $\nu = \gamma$ **else** $\mu = \gamma$
22: **end for**
23: ($\mathcal{D}$, r')= k-COVER(P, k, ν) //Run Algorithm 5.4
24: **Return** ($\mathcal{D}$, r')

$\epsilon) r_{opt}$. Line 23 in Algorithm 5.5 invokes Algorithm 5.4, which returns a set $\mathcal{D}$ of k disks of radius r', centered on ∂P, where $r' \leq (1 + \frac{7}{k})\nu$ (see Theorem 5.1.13). Hence, $r' \leq (1 + \frac{7}{k})\nu \leq (1 + \frac{7}{k})(1 + \epsilon) r_{opt} \leq (1 + \frac{7}{k} + \frac{7\epsilon}{k} + \epsilon) r_{opt}$. Algorithm 5.4 is invoked at most $|\log r_{opt}|$ times in the while-loop and at most $\log \lceil \frac{1}{\epsilon} \rceil$ times in the for-loop at lines 6 or 12 and 18 of Algorithm 5.5, respectively. The running time of Algorithm 5.4 is $O(n(n + k))$ (see Lemma 5.1.12). Hence, the running time of Algorithm 5.5 is $n(n + k)|\log r_{opt}| + n(n + k) \log \lceil \frac{1}{\epsilon} \rceil = O(n(n + k)(|\log r_{opt}| + \log \lceil \frac{1}{\epsilon} \rceil))$. □

5.3 Conclusion

In this chapter we have described an approximation algorithm for covering a convex polygon with k congruent disks centered on the boundary of the polygon. The approximation factor of the algorithm is $(1 + \frac{7}{k} + \frac{7\epsilon}{k} + \epsilon)$, where $k \geq 7$ and $\epsilon > 0$. The approximation factor of the previous best known algorithm is 1.8841 [25]. Thus, for a sufficiently large value of k, our algorithm is much better than the previous one. The complexity of this problem seems to be unknown as we did not find any prior work on the NP-hardness proof of this problem. As future work, designing a $(1 + \epsilon)$-approximation algorithm for every $\epsilon > 0$ (PTAS), can be considered and the hardness of the problem can be explored.

Chapter 6

The Euclidean k-Supplier Problem

In this chapter we consider the k-supplier problem in $\mathbb{R}^2$, as follows:

> Two sets of points $\mathcal{P}$ and $\mathcal{Q}$ are given. The objective is to choose a subset $\mathcal{Q}_{opt} \subseteq \mathcal{Q}$ of size at most k such that the union of the congruent disks of minimum radius centered at the points in $\mathcal{Q}_{opt}$ covers all the points of $\mathcal{P}$.

The k-supplier problem has been very well studied in the literature. The previous best solution provides a $(1+\sqrt{3})$-approximation in polynomial time, but the problem is known to be NP-hard to approximate beyond $\sqrt{7}$. We present in this chapter a 2-factor approximation algorithm, which goes beyond the lower bound ($<\sqrt{7}$), but runs in time exponential in k. Our algorithm relies on very simple techniques, and it can further be improved by refining the same simple techniques. In this chapter, we also present a heuristic algorithm based on iteratively computing the Voronoi diagrams. We analyze the running time of each iteration of our heuristic algorithm, but we neither bound the overall running time nor the approximation factor. Hence, we evaluate our heuristic algorithm experimentally against the previous best known algorithm and show that our algorithm is little slower, but achieves better approximation.

We first propose a fixed-parameter tractable (FPT) algorithm for the k-supplier problem that produces a 2-approximation result. For $|\mathcal{P}| = n$ and $|\mathcal{Q}| = m$, the worst case running time of the algorithm is $O(6^k(n+m)\log(mn))$, which is an exponential function of the parameter k. We also generalize the idea for developing a FPT 2-approximation algorithm and propose FPT $(1+\epsilon)$-approximation algorithm for the

k-supplier problem in the plane, where $\epsilon > 0$ is an arbitrary constant. The running time of the $(1+\epsilon)$-approximation algorithm is $O(\epsilon^{-2k}(m+n)\log(mn))$. Similarly, for the Euclidean k-supplier problem in $\mathbb{R}^d$, we obtain an FPT $(1+\epsilon)$-approximation algorithm for any arbitrary constant $\epsilon > 0$. The running time of the $(1+\epsilon)$-approximation algorithm is $O(\epsilon^{-dk}(m+n)\log(mn))$. We also propose a heuristic algorithm based on the Voronoi diagram for the k-supplier problem, and experimentally compare the result produced by the proposed algorithm with the best known approximation algorithm available in the literature. The experimental results show that our heuristic algorithm outperforms the best known existing result.

In Section 6.1, we discuss FPT approximation algorithms for the Euclidean k-supplier problem. In the same section, we also propose $(1+\epsilon)$-approximation algorithms. In Section 6.2, we present a heuristic algorithm and its theoretical as well as experimental performance with the existing result for the k-supplier problem in $\mathbb{R}^2$. Finally, we conclude the chapter in Section 6.3.

6.1 FPT 2-Approximation Algorithm

6.1.1 Terminologies

Let $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ and $\mathcal{Q} = \{q_1, q_2, \ldots, q_m\}$ denote a set of n clients and a set of m facilities respectively in $\mathbb{R}^2$. Throughout the chapter we use $\delta(a,b)$ to denote the Euclidean distance between a pair of points $a, b \in \mathbb{R}^2$, and $\Delta(a,r)$ to denote the region covered by the disk of radius r centered at point a. Let $D = \{\delta(p,q) \mid p \in \mathcal{P} \text{ and } q \in \mathcal{Q}\}$ be the set of distances between the points in $\mathcal{P}$ and the points in $\mathcal{Q}$. Let $r_1, r_2, \ldots, r_{mn}$ be the non-decreasing order of the members in D. Let $\mathcal{Q}_{opt}$ be an optimal solution of the k-supplier problem, and r_{opt} be the radius of the disks in $\mathcal{Q}_{opt}$.

Lemma 6.1.1. $r_{opt} \in \{r_1, r_2, \ldots, r_{mn}\}$.

Proof. Assume that $r_{opt} \notin \{r_1, r_2, \ldots, r_{mn}\}$. Then there must exist an i such that $r_i < r_{opt} < r_{i+1}$. Thus, no point in $\mathcal{P}$ lies on the boundary of any of the disks in the optimal solution centered at k points in $\mathcal{Q}$. Therefore, we can reduce the radius of every disk and still cover $\mathcal{P}$. This contradicts the fact that r_{opt} is the minimum radius. Thus the lemma follows. $\square$

6.1.2 Approximation Algorithm

In this subsection we propose a parameterized 2-approximation algorithm for k-supplier problem. For a given instance $(\mathcal{P}, \mathcal{Q})$ of the k-supplier problem, the objective is to choose a subset $\hat{\mathcal{Q}} \subseteq \mathcal{Q}$ of size at most k such that the union of k disks of radius $r \leq 2r_{opt}$ centered at the points in $\hat{\mathcal{Q}}$ covers all the points in $\mathcal{P}$.

Let us first consider the following decision problem:

> For a given radius r, does there exist a subset $\mathcal{Q}' \subseteq \mathcal{Q}$ of size at most k (i.e. $|\mathcal{Q}'| \leq k$) such that the union of k disks of radius $2r$ centered at the points of $\mathcal{Q}'$ covers all the points in $\mathcal{P}$?

We show that the above decision problem can be solved with time complexity $O(\alpha^k \texttt{polynomial}(m, n))$, where α is a predefined constant. For a given radius r, if the answer is positive, then it reports the chosen subset $\mathcal{Q}'$ with $true$. For a negative reply, it reports the chosen subset $\mathcal{Q}'$ with $false$ (wrong $\mathcal{Q}'$).

We apply binary search in the set $D = \{r_1, r_2, \ldots, r_{mn}\}$ to find the minimum r for which the above decision problem returns a positive reply (i.e. a subset $\mathcal{Q}' \subseteq \mathcal{Q}$, where $|\mathcal{Q}'| \leq k$).

Let the point $p \in \mathcal{P}$ be covered by a disk of radius r centered at $q \in \mathcal{Q}$. Thus $q \in \Delta(p, r)$. Let us draw six radii of the circular region $\Delta(p, r)$ such that each pair of consecutive radii make an angle $\frac{\pi}{3}$ at the point p. These split the region $\Delta(p, r)$ into six equal sectors $\Delta^1, \Delta^2, \ldots, \Delta^6$ as shown in Figure 6.1.

Lemma 6.1.2. *If $q \in \Delta^i, i \in \{1, 2, \ldots, 6\}$, then any disk of radius $2r$ centered at any point in the region Δ^i will cover all the points of $\mathcal{P}$ that are covered by the disk of radius r centered at q.*

Proof. Follows from the triangle inequality and the facts that (i) $\delta(p', q) \leq r$ for any point $p' \in \mathcal{P}$ that is covered by the disk of radius r centered at q, and (ii) $\delta(q, q') \leq r$ for any point $q' \in \mathcal{Q} \cap \Delta^i$. $\square$

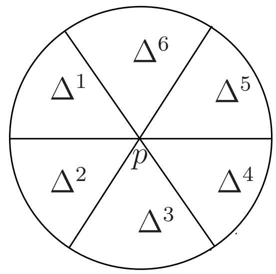


Figure 6.1: Partition of the disk $\Delta(p, r)$ into six equal sectors $\Delta^1, \Delta^2, \ldots, \Delta^6$

We solve the decision problem for a given radius r as follows. We initialize $\mathcal{P}' = \mathcal{P}$ and $\mathcal{Q}' = \emptyset$. Choose a point $p \in \mathcal{P}'$, and arbitrarily partition the disk $\Delta_1 = \Delta(p, r)$ into six equal sectors $\Delta_1^1, \Delta_1^2, \ldots, \Delta_1^6$ as shown in Figure 6.1.

We consider each sector $\Delta_1^i, i = 1, 2, \ldots, 6$ separately. For each sector Δ_1^i, if $\mathcal{Q} \cap \Delta_1^i \neq \emptyset$, then we can choose any point $q \in \mathcal{Q} \cap \Delta_1^i$ (by Lemma 6.1.2). Update $\mathcal{Q}'$ by $\mathcal{Q}' \cup \{q\}$. Let $\mathcal{R} \subseteq \mathcal{P}$ be the set of points lying in $\Delta(q, 2r)$. We update $\mathcal{P}' = \mathcal{P}' \setminus \mathcal{R}$. If the updated $\mathcal{P}' \neq \emptyset$, we repeat the same process recursively to find $q' \in \mathcal{Q}$ by arbitrarily choosing a point $p' \in \mathcal{P}'$ (updated), drawing $\Delta_2^i, i = 1, 2, \ldots, 6$, and then processing each Δ_2^i to update $\mathcal{Q}' = \mathcal{Q}' \cup \{q'\}$. The process along a path of the recursion stops if either the updated $\mathcal{P}'$ up to that level of recursion is empty, or the level of recursion is k.

Thus, our search process progresses in a tree like fashion, where the degree of each node in this search tree is at most six, and the maximum length from the root up to a leaf in any search path is at most k. At the end of this process, if the resulting set $\mathcal{P}' = \emptyset$ along any one of the search path explored, then we return the corresponding $\mathcal{Q}'$ with $true$, otherwise we return $\mathcal{Q}'$ with $false$. Thus after executing this decision algorithm, it indicates a positive answer if return value is $true$ with $\mathcal{Q}'$, and a negative answer if return value is $false$ with $\mathcal{Q}'$.

The pseudocode of the procedure for computing the minimum r such that at most k disks of radius $2r$ centered at the points in $\mathcal{Q}$ covers all the points in $\mathcal{P}$ is given in Algorithm 6.1.

Lemma 6.1.3. *If Algorithm 6.2 is invoked with $r = r_{opt}$, then it produces a positive reply. In other words, it produces a subset $\hat{\mathcal{Q}} \subseteq \mathcal{Q}$ of size at most k such that union of the disks with radius $2r$ centered at the points in $\hat{\mathcal{Q}}$ covers all the points in $\mathcal{P}$.*

Algorithm 6.1 k-supplier($\mathcal{P}, \mathcal{Q}, k$)

Input: A set $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ of n points, a set $\mathcal{Q} = \{q_1, q_2, \ldots, q_m\}$ of m points, and a positive integer k.
Output: a set of points $\hat{\mathcal{Q}} \subseteq \mathcal{Q}$ of size at most k and a radius $\hat{r}$ such that the union of the disks with radius $2\hat{r}$ centered at the points in $\hat{\mathcal{Q}}$ covers all points in $\mathcal{P}$.
Let $\{r_1, r_2, \ldots, r_{mn}\}$ be the elements of $\{\delta(p, q) \mid p \in \mathcal{P}\ \&\ q \in \mathcal{Q}\}$ in non-decreasing order.
$\ell \leftarrow 1$; $h \leftarrow mn$;
while $(\ell < h)$ **do**
    $mid = \lceil \frac{\ell+h}{2} \rceil$; $\mathcal{P}' \leftarrow \mathcal{P}$; $\mathcal{Q}' \leftarrow \emptyset$; /* $\mathcal{Q}'$ will hold at most k centers */
    $(flag, \mathcal{Q}') =$ **cover**$(\mathcal{P}', \mathcal{Q}, k, r_{mid})$ /* Invoke Algorithm 6.2 */
    if $(flag)$ **then**
        $\hat{\mathcal{Q}} = \mathcal{Q}'$; $h = mid - 1$; $\hat{r} = r_{mid}$;
    else
        $\ell = mid + 1$
    end if
end while
return $(\hat{\mathcal{Q}}, \hat{r})$

Proof. Note that, in the optimum solution, each member of $\mathcal{P}$ is covered by some element of $\mathcal{Q}_{opt}$. In Algorithm 6.2, suppose we have chosen some element $p_1 \in \mathcal{P}$, and its corresponding covering element $q \in \mathcal{Q}_{opt}$ lies in Δ_1^i of $\Delta(p_1, r_{opt})$. Let $\mathcal{P}_1 = \mathcal{P} \setminus (\mathcal{P} \cap \Delta(q, r_{opt}))$. Now, in the 6-way search tree of Algorithm 6.2, if we proceed towards a point $\hat{q} \in \Delta_1^i$ then the remaining uncovered points $\mathcal{P}_1' = \mathcal{P} \setminus (\mathcal{P} \cap \Delta(\hat{q}, 2r_{opt})) \subseteq \mathcal{P}_1$ (see Lemma 6.1.2). Again, for any element $p_2 \in \mathcal{P}_1'$, if its covering element is $q' \in \mathcal{Q}_{opt}$, and in the 6-way search tree if we proceed with a point $\hat{q}'$ in the sector of $\Delta(p_2, r_{opt})$ containing q', the remaining uncovered points $\mathcal{P}_2' = \mathcal{P} \setminus (\mathcal{P} \cap (\Delta(\hat{q}, 2r_{opt}) \cup \Delta(\hat{q}', 2r_{opt})))$ will be a subset of $\mathcal{P}_2 = \mathcal{P} \setminus (\mathcal{P} \cap (\Delta(q, r_{opt}) \cup \Delta(q', r_{opt})))$ (see Lemma 6.1.2). The maximum depth along such a search path will be less than or equal to k since $|\mathcal{Q}_{opt}| \leq k$. As we are exploring all possible search paths, the result follows. □

Lemma 6.1.4. *If $\hat{\mathcal{Q}}$ and r are the output of Algorithm 6.1, then the union of the disks with radius $2r$ centered at the points in $\hat{\mathcal{Q}}$ covers all the points in $\mathcal{P}$ and $r \leq r_{opt}$.*

Proof. Let $\mathcal{P}, \mathcal{Q}, k, r$ be the input of Algorithm 6.2 (**cover**). Lemma 6.1.3 says that if $r = r_{opt}$ then the Algorithm 6.2 returns a subset $\mathcal{Q}' \subseteq \mathcal{Q}$ such that the union of the disks with radius $2r$ centered at the points in $\mathcal{Q}'$ covers all the points in $\mathcal{P}$. Lemma 6.1.1 says

Algorithm 6.2 cover($\mathcal{P}, \mathcal{Q}, k, r$)

Input: The set $\mathcal{P}$ of uncovered points, a set $\mathcal{Q}$ of m facility points, a positive integer k and a radius r.
Output: (i) $true$ if cover of $\mathcal{P}$ is achieved with at most k disks of radius $2r$; $false$ otherwise and (ii) a set $\mathcal{Q}'$ to hold centers of at most k disks.
if $(\mathcal{P} = \emptyset)$ **then**
    return $(true, \emptyset)$
else if $(k = 0)$ **then**
    return $(false, \emptyset)$
else
    Consider the disk $\Delta(p, r)$ centered at an arbitrary point $p \in \mathcal{P}$, and
    partition $\Delta(p, r)$ into six equal sectors $\Delta^1, \Delta^2, \ldots, \Delta^6$ as shown in Figure 6.1.
    $i \leftarrow 0$; $flag \leftarrow false$; $\mathcal{Q}' \leftarrow \emptyset$
    while ($i < 6$ and $flag$=$false$) **do**
        if $(\mathcal{Q} \cap \Delta^i \neq \emptyset)$ **then**
            choose a point $q \in \mathcal{Q} \cap \Delta^i$
            $\mathcal{P}' = \mathcal{P} \setminus (\mathcal{P} \cap \Delta(q, 2r))$;
            $(flag, \mathcal{Q}')$ = **cover**$(\mathcal{P}', \mathcal{Q}, k-1, r)$
            if ($flag$=$true$) **then**
                $\mathcal{Q}' = \mathcal{Q}' \cup \{q\}$
            end if
        end if
        $i \leftarrow i + 1$;
    end while
end if
return $(flag, \mathcal{Q}')$

that $r_{opt} \in \{r_1, r_2, \ldots, r_{mn}\}$. Algorithm 6.1 finds a minimum value $\hat{r} \in \{r_1, r_2, \ldots, r_{mn}\}$ such that with the input $\mathcal{P}, \mathcal{Q}, k, \hat{r}$, the Algorithm 6.2 returns a set $\hat{\mathcal{Q}} \subseteq \mathcal{Q}$ such that the union of the disks with radius $2\hat{r}$ centered at the points in $\hat{\mathcal{Q}}$ covers all the points in $\mathcal{P}$. Therefore $\hat{r} \leq r_{opt}$. □

Theorem 6.1.5. *Algorithm 6.1 for the k-supplier problem produces a 2-approximation result, and it runs in $O(6^k(n+m)\log(nm))$ time.*

Proof. Approximation factor follows from Lemma 6.1.4.

Let $T(n, m, k)$ be the running time of the Algorithm 6.2, where $n = |\mathcal{P}|$, $m = |\mathcal{Q}|$. Since each node of the recursion tree has degree at most 6, and execution at that node takes $O(n+m)$ time, we have $T(n, m, k) = 6((m+n) + T(n, m, k-1))$. Again, since the depth of the recursion tree is at most k, we have the running time of the Algorithm 6.2 is $O(6^k(n+m))$. Algorithm 6.1 invokes the Algorithm 6.2 at most $\log(mn)$ times. Thus the total running time is $O(6^k(n+m)\log(mn))$.

□

Corollary 6.1.6. *For $k = O(\log n)$, the k-supplier problem in $\mathbb{R}^2$ has a 2-approximation algorithm that runs in polynomial time.*

We can extend the idea of solving the k-supplier problem in $\mathbb{R}^2$ to solve the k-supplier problem in $\mathbb{R}^3$. Here, we consider a ball of radius r in $\mathbb{R}^3$ instead of a disk of radius r in $\mathbb{R}^2$. We partition the ball of radius r into 12 equal sectors such that the distance between any two points in a sector is at most $\sqrt{2}r$. The remaining part of the algorithm for $\mathbb{R}^3$ is exactly the same as in the case of $\mathbb{R}^2$. Thus we have the following theorem.

Theorem 6.1.7. *For the k-supplier problem in $\mathbb{R}^3$, we can get a $(1+\sqrt{2})$-approximation result in $O(12^k(n+m)\log(mn))$ time.*

To get finer approximations (less than 2), we can generalize the same technique used for developing the above FPT approximation algorithms as follows: In the 6-way search algorithm (in Algorithm 6.2), every time we center a disk $\Delta(p,r)$ at an arbitrary point $p \in \mathcal{P}$, we partition $\Delta(p,r)$ into $O(\frac{1}{\epsilon^2})$ sectors such that the distance between any two arbitrary points within any of these sectors is at most ϵr. Then, we place a disk $\Delta(q, r+\epsilon r)$ centered at a point $q \in \mathcal{Q}$ instead of $\Delta(q, 2r)$ (lines 13 and 14 in Algorithm 6.2). As a result, we obtain FPT $(1+\epsilon)$-approximation algorithm for the k-supplier problem in $\mathbb{R}^2$, where $\epsilon > 0$ is an arbitrary constant. The running time of FPT $(1+\epsilon)$-approximation algorithm is $O(\epsilon^{-2k}(m+n)\log(mn))$. Let V be the d-dimensional volume of a Euclidean ball of radius r in $\mathbb{R}^d$. We know that $V = O(r^d)$. For any constant $\epsilon > 0$, if v is the d-dimensional volume of a Euclidean ball of radius $\frac{\epsilon r}{2}$ in $\mathbb{R}^d$, then $v = O((\epsilon r)^d)$. The number of balls of radius $\frac{\epsilon r}{2}$ required to cover a ball of radius r in $\mathbb{R}^d$ is $\frac{V}{v} = O(\epsilon^{-d})$. Now observe that the distance between any two arbitrary points lying in each of smaller balls of radius $\frac{\epsilon r}{2}$ is at most ϵr, so we treat these smaller balls as sectors of a ball of radius r in $\mathbb{R}^d$. Hence, we make the following remark.

Remark 6.1.8. The Euclidean k-supplier problem in $\mathbb{R}^d$ admits an FPT $(1+\epsilon)$-approximation algorithm, and the running time of the FPT $(1+\epsilon)$-approximation algorithm is $O(\epsilon^{-dk}(m+n)\log(mn))$, where $\epsilon > 0$ is a constant and d is a positive integer.

Note: It needs to be mentioned that in the Algorithm 6.2, instead of splitting the disk $\Delta(p, r)$ in 6 parts, and choosing points q in these parts, if we choose the point p itself and reduce the set of uncovered points by $\Delta(p, 2r)$, then we can generate a 2-approximation result for the unconstrained (general) version of the k-center problem in time $O(kn + n^2 \log n)$ time.

6.2 Heuristic Algorithm for the k-Supplier Problem

In this section we present a heuristic algorithm for the k-supplier problem in $\mathbb{R}^2$ based on the Voronoi diagram [7]. Initially, we pick a set of k arbitrary points $\mathcal{Q}' = \{q'_1, q'_2, \ldots, q'_k\} \subseteq \mathcal{Q}$. We compute a nearest point Voronoi diagram $VOR(\mathcal{Q}')$ of the points in $\mathcal{Q}'$. This forms the clusters of the points in $\mathcal{P}$, namely $\mathcal{P}_i = \{p \in \mathcal{P} \mid p \in vor(q'_i)\}$, where $vor(q'_i)$ is the Voronoi cell of the point $q'_i \in \mathcal{Q}'$, $i = 1, 2, \ldots, k$. For each cluster $\mathcal{P}_i$, let d_i be the smallest disk centered at a point in $\mathcal{Q}$ such that $\mathcal{P}_i \subset d_i$. Let $r = \max(r_1, r_2, \ldots, r_k)$, where r_i is the radius of the disk d_i, and $\mathcal{Q}'' = \{q''_i \mid q''_i$ is the center of the disk d_i, $i = 1, 2, \ldots, k\}$. We repeat this process by setting $\mathcal{Q}' = \mathcal{Q}''$. The process continues as long as the radius r decreases. The detailed pseudocode of this procedure is given in Algorithm 6.3.

Lemma 6.2.1. *At each iteration of Algorithm 6.3 the value of r (radius of congruent disks) never increases.*

Proof. Let $q_1, q_2, \ldots, q_k$ be the Voronoi sites (cluster centers) at the beginning of an iteration. The minimum enclosing disks of $\mathcal{P} \cap vor(q_1), \mathcal{P} \cap vor(q_2), \ldots, \mathcal{P} \cap vor(q_k)$, centered at the points in $\mathcal{Q}$, are $d'_1, d'_2, \ldots, d'_k$ respectively. Let the center of the disk d'_i be q'_i and the radius be r'_i, $i = 1, 2, \ldots, k$.

Now consider an arbitrary Voronoi cell $vor(q_i)$ $(1 \leq i \leq k)$. Without loss of generality assume that $vor(q_1), vor(q_2), \ldots, vor(q_t)$ are the neighboring Voronoi cells of $vor(q_i)$. We consider the following cases: (a) $r'_i \geq \max\{r'_1, r'_2, \ldots, r'_t\}$, and (b) $r'_i < \max\{r'_1, r'_2, \ldots, r'_t\}$.

In this iteration, let d''_i denote the minimum enclosing disk of the points in $\mathcal{P} \cap vor(q'_i)$, whose center is at a point $q''_i \in \mathcal{Q}$, and the radius is r''_i, $i = 1, 2, \ldots, k$. Here, $d''_i \neq d'_i$ only if $\mathcal{P} \cap vor(q'_i) \neq \mathcal{P} \cap vor(q_i)$.

Case (a): It is sufficient to show that $r''_i \leq r'_i$. On the contrary, assume that $r''_i > r'_i$. Therefore there exists at least one point $p \in \mathcal{P} \cap vor(q_s)$ (for some $s \in \{1, 2, \ldots, t\}$) in the previous iteration, but $p \in \mathcal{P} \cap vor(q'_i)$ in this iteration. We choose p to be the one which is farthest from q'_i among the points which entered from some other Voronoi cell to $vor(q'_i)$ in this iteration. The Voronoi partitioning suggests that $\delta(q'_i, p) \leq \delta(q'_s, p)$.

Again, observe that $r''_i \leq \delta(q'_i, p)$ since r''_i is the radius of the minimum enclosing disk d''_i containing all the points $\mathcal{P} \cap vor(q'_i)$ and its center q''_i may be different from q'_i. Since q'_s is the center of the minimum enclosing disk (centered at a point in $\mathcal{Q}$) for a point set containing p, we have $\delta(q'_s, p) \leq r'_s$. Again, $r'_s \leq r'_i$ by our assumption. Thus we have, $r''_i \leq r'_i$, which leads to a contradiction.

Case (b): The lemma is trivial if $r''_i \leq r'_i$. Therefore we assume that $r''_i > r'_i$. Thus, we have at least one point $p \in \mathcal{P} \cap vor(q_s)$ (for some $s \in \{1, 2, \ldots, t\}$) such that p moves to the cell $vor(q'_i)$ in this iteration. From the properties of a Voronoi partition $\delta(q'_i, p) \leq \delta(q'_s, p)$. As in the former case, we have $r''_i \leq \delta(q'_i, p)$ and $r'_s \geq \delta(q'_s, p)$. Therefore, $r''_i \leq \delta(q'_i, p) \leq \delta(q_s, p) \leq r'_s$ i.e., $r''_i \leq r'_s$. Thus, the lemma follows in this case also.

□

Lemma 6.2.2. *The worst case time complexity of every iteration in Algorithm 6.3 is* $O((m + n)\log(nk))$.

Proof. In the **while** loop, computing a Voronoi diagram (line 7) for a set of k points takes $O(k \log k)$ time. All the clusters $\mathcal{P}_i$ $(i = 1, 2, \ldots, k)$ can be computed (lines 9-11) in $O(n \log k)$ time using planar point location in $VOR(\mathcal{Q}')$. In order to compute the new cluster centers, for each point $q \in \mathcal{Q}$ first identify in which Voronoi cell it falls (line 17 in the **for** loop at line 16), and then locate its furthest neighbor (line 18 in the **for** loop at line 16) among the vertices of the convex hull of $\mathcal{P}_i$. This needs $O(m(\log k + \log n_i))$ time, where $n_i = |CH(\mathcal{P}_i)|$. The computation of $CH(\mathcal{P}_i)$ and $FVD(\mathcal{P}_i)$ for all $i = 1, 2, \ldots, k$ needs $O(n \log n)$ time. Thus, the overall time complexity for a single iteration is $O((m + n)\log(nk))$. □

6.2.1 Experimental Results

We implemented our Voronoi diagram based heuristic algorithm, and the best known algorithm available in the literature for the k-supplier problem in $I\!R^2$ [69]. We have used Matlab on a machine equipped with an Intel Pentium(R) CPU G870 @ 3.10GHz × 2, 1.8 GB RAM and running 64-bit Ubuntu 12.03 to implement both the algorithms. We have chosen two sets of points within a square 20 times for different values of m, n and k. We executed the algorithms for each chosen instance and finally returned average values as output (see Table 6.1). Radii of disks for our Voronoi diagram based heuristic algorithm and for the algorithm in [69] are denoted by r_{vor} and r_{nag} respectively and the execution times (in second) of the algorithms are denoted as t_{vor} and t_{nag} respectively. The computed radii results indicate that our Voronoi diagram based heuristic algorithm produces a much better result than the algorithm in [69]. However, the execution time of our algorithm is little more than that of [69].

Algorithm 6.3 k-Supplier-Heuristic($\mathcal{P}, \mathcal{Q}, k$)

Input: A set $\mathcal{P}$ of n points, a set $\mathcal{Q}$ of m points, and a positive integer k.
Output: A set $\mathcal{D}$ of k disks of radius r centered at k points of $\mathcal{Q}$ such that $\mathcal{P} \subseteq \bigcup_{d \in \mathcal{D}} d$.
$r_{old} \leftarrow \infty$; $r_{new} = \max\{\delta(p, q) \mid p \in \mathcal{P} \ \& \ q \in \mathcal{Q}\}$
Let $\mathcal{Q}' = \mathcal{Q}'' = \{q_1, q_2, \ldots, q_k\} \subseteq \mathcal{Q}$ be an arbitrary subset of k points, called *cluster centers*.
while ($r_{new} < r_{old}$) **do**
  $r_{old} = r_{new}$, $\mathcal{Q}' = \mathcal{Q}''$
  Compute $VOR(\mathcal{Q}')$.
  (* Compute the cluster $\mathcal{P}_i = \mathcal{P} \cap vor(q_i)$ *)
  for $i = 1, 2, \ldots, n$ **do**
    Apply point location with the point p_i in $VOR(\mathcal{Q}')$ to assign it to an appropriate cluster.
  end for
  for $i = 1, 2, \ldots, k$ **do**
    Compute the convex hull $CH(\mathcal{P}_i)$ of the points in $vor(q_i)$, and the furthest point Voronoi diagram $FVD(\mathcal{P}_i)$ of the vertices of $CH(\mathcal{P}_i)$.
  end for
  Create two arrays, namely $r[1, 2 \ldots, k]$ initialized with $[\infty, \infty, \ldots, \infty]$ and $\mathcal{Q}''[1, 2, \ldots, k]$ to store the new *cluster centers*.
  for $j = 1, 2, \ldots, m$ **do**
    Find i such that $q_j \in vor(q_i)$
    Consult $FVD(\mathcal{P}_i)$ to find a vertex p of $CH(\mathcal{P}_i)$ which is furthest from q_j among the other vertices of $CH(\mathcal{P}_i)$.
    compute $\delta(p, q_j)$;
    if $\delta(p, q_j) < r(\mathcal{P}_i)$ **then**
      Assign $r[i] = \delta(p, q_j)$; $\mathcal{Q}''[i] = q_j$
    end if
  end for
  (* For each i, the minimum enclosing disk d_i for the cluster $\mathcal{P}_i$ is centered at $\mathcal{Q}''[i]$ and has radius $r[i]$. *)
  Compute $r_{new} = \max\{r[1], r[2], \ldots, r[k]\}$
end while
$r = r_{old}$, $\mathcal{D} = \emptyset$
for ($i = 1, 2, \ldots, k$) **do**
  Let d_i be the disk of radius r centered at $q_i \in \mathcal{Q}'$
  $\mathcal{D} = \mathcal{D} \cup \{d_i\}$
end for
Return $(r, \mathcal{D})$

n	m	k	r_{vor}	r_{nag}	t_{vor}	t_{nag}
100	50	20	385.1651	767.8174	0.0204	0.0052
200	100	50	331.8947	474.1622	0.0536	0.0148
500	400	50	271.5503	354.4516	0.4824	0.0767
500	400	100	187.3902	330.0519	0.9357	0.0883
500	400	200	165.4990	275.2376	1.8368	0.0871
500	400	300	157.4919	352.6021	1.8220	0.0758
800	400	100	220.0354	348.7830	1.7462	0.1114
800	400	200	162.2155	299.2550	2.5599	0.1308
800	400	300	188.4023	431.0273	2.5415	0.1003
800	600	100	226.4105	315.7258	1.9460	0.1540
800	600	200	183.6834	347.9941	3.8523	0.1372
800	600	300	153.9939	287.3068	5.7032	0.1638
800	600	400	162.6617	417.0634	5.0574	0.1249
800	600	500	129.5877	290.8249	6.2892	0.1639
800	700	100	279.8605	313.5049	1.5080	0.1685
800	700	200	168.2815	248.6119	4.4462	0.2011
800	700	300	145.6643	279.1926	6.6276	0.1942
800	700	400	134.3790	337.201	5.9084	0.1706
800	700	500	139.7529	224.3580	7.5274	0.2149
800	700	600	156.6618	239.7690	8.7611	0.2089
1000	800	100	190.4600	243.5794	4.1017	0.2829
1000	800	200	158.9247	256.8944	6.0269	0.2598
1000	800	300	186.3586	253.9525	9.1140	0.2611
1000	800	400	120.5850	246.3848	11.9620	0.2714
1000	800	500	113.2485	228.0754	10.0219	0.2947
1000	800	600	134.3641	266.7492	11.9123	0.2483
1000	800	700	100.2370	248.2916	14.0810	0.2705
1000	900	100	226.1000	300.7388	4.6818	0.2506
1000	900	200	143.3740	252.6710	9.0590	0.2932
1000	900	300	133.1333	229.2618	10.1513	0.3144
1000	900	400	122.3652	293.7928	13.5053	0.2568
1000	900	500	113.3674	238.9505	16.8278	0.3009
1000	900	600	108.2268	261.4600	18.1648	0.2833
1000	900	700	101.3077	217.5035	20.4861	0.3194
1000	900	800	100.5504	230.9309	22.9282	0.3095

Table 6.1: Radii of disks centered by different algorithms and their execution time

6.3 Conclusion

In this chapter we have proposed a fixed-parameter tractable (FPT) algorithm for the k-supplier problem in $\mathbb{R}^2$. Our proposed FPT algorithm produces a 2-approximation result. The running time of the proposed FPT algorithm is $O(6^k(n+m)\log(nm))$, where k is the parameter. The proposed FPT algorithm can be extended to $\mathbb{R}^3$, and we can get a $(1+\sqrt{2})$-approximation result with running time $O(12^k(n+m)\log(nm))$. We have also shown that this FPT algorithm can be generalized to get a $(1+\epsilon)$-approximation algorithm, which runs in $O(\epsilon^{-dk}(n+m)\log(nm))$ time, for a d-dimensional Euclidean space. For the k-supplier problem in $\mathbb{R}^2$, we have also proposed a heuristic algorithm based on Voronoi diagram. We did an experimental study on this heuristic algorithm and compared it with the best known algorithm available in the literature for the k-supplier problem in $\mathbb{R}^2$. Experimental results indicate that the heuristic algorithm performs better than the best known algorithm available in the literature with very minor degradation in the running time.

Chapter 7

Conclusions and Future Works

In this thesis we have investigated various types of geometric covering problems such as *line separable discrete unit disk cover* (LSDUDC) problem, *discrete unit disk cover* (DUDC) problem, *rectangular region cover* (RRC) problem, *rectangular region cover* problem in reduce radius setup, *strip square cover* (SSC) problem, *discrete unit square cover* (DUSC) problem, *constrained convex polygon cover* (CCPC) problem, and the *Euclidean k-supplier* problem. Most of these problems are NP-hard and hardness proofs of remaining problems were not found in the literature, so are open for investigation. Numerous approximation algorithms, exhibiting trade-offs between approximation factors and time complexities, have already been proposed for these problems in the literature. In this thesis, the objective has been to provide approximation algorithms with improved approximation factor or improved time complexity or both over the previous best known algorithms available in the literature.

For the LSDUDC problem we have proposed an $(1 + \mu)$-approximation algorithm i.e. a *polynomial time approximation scheme* (PTAS), where $0 < \mu \leq 1$. The running time of our proposed PTAS is $O(m^{3(1+\frac{1}{\mu})} n \log n)$. Using this PTAS, we proposed a $(9 + \epsilon)$-approximation algorithm for the DUDC problem, which improved the previous 15-approximation result for the same problem [33], where $0 < \epsilon \leq 6$. The running time of our proposed $(9 + \epsilon)$-approximation algorithm for the DUDC problem is $O(m^{3(1+\frac{6}{\epsilon})} n \log n)$. We have also proposed a $(9+\epsilon)$-approximation algorithm for the RRC problem, where $0 < \epsilon \leq 6$. The running time of the proposed algorithm for the RRC problem is $O(m^{5+\frac{18}{\epsilon}} \log m)$. In the reduce radius setup we proposed a PTAS. We have proposed a 2.25-approximation algorithm for the RRC problem in reduce radius setup using the proposed PTAS result. The previous best known approximation factor for the RRC problem in reduce radius setup was 4 [32]. The running time of our proposed algorithm for the RRC problem in reduce radius setup is less than that of 4-approximation algorithm proposed in [32] for reasonably small value of $\gamma(= \frac{\nu}{\sqrt{2}})$, where γ is the radius reduction parameter. Hence, our result for the RRC problem in reduce radius setup is a significant improvement over the previous result, in terms of both approximation factor as well as time complexity. However, the LSDUDC problem is still open, in the sense that there is neither an algorithm which solves LSDUDC optimally in polynomial time, nor is there a known polynomial time reduction from any NP-hard problem to LSDUDC problem. Although the RRC problem seems to be NP-hard, we did not find the NP-hardness proof of the problem in the literature. Therefore, the complexity of the RRC problem is also open.

For the DUSC problem we have proposed $(2 + \frac{4}{k-2})$-approximation algorithm in $O(km^k n)$ time, where $k(> 2)$ is an integer. The time complexity of our proposed approximation algorithm is faster than the best known algorithm available in the literature for $k \in \{5, 6, \ldots, 8\}$ by sacrificing some approximation factor [78]. Our solution of the DUSC problem is based on a simple $(1 + \frac{2}{k-2})$-approximation algorithm for the SSC problem, where $k > 2$. We also developed an algorithm to solve the SSC problem optimally. Our proposed algorithm is based on plane sweep and graph search traversal techniques. The running time of the algorithm is $O(m^4 n + n \log n)$. Using this result for the SSC problem, we presented a 2-approximation algorithm for the DUSC problem. The running time of the proposed 2-approximation algorithm for the DUSC problem is $O(m^4 n + n \log n)$, which is an improvement by a factor of $O(m^4 n)$ over the 2-approximation algorithm [62]. As future work, we can consider developing an algorithm with better running time for the SSC problem and also designing a better approximation algorithm for the DUSC problem.

In our solution of the CCPC problem we have described an algorithm for covering a convex polygon P with k congruent disks of radius $r \leq (1 + \frac{7}{k} + \frac{7\epsilon}{k} + \epsilon) r_{opt}$, centered on the boundary of the polygon P, where r_{opt} is the optimum radius of k congruent disks,

$k \geq 7$ and an $\epsilon > 0$. Hence, the approximation factor of the proposed algorithm for the CCPC problem is $(1 + \frac{7}{k} + \frac{7\epsilon}{k} + \epsilon)$. The running time of the proposed approximation algorithm is $O(n(n+k)(|\log r_{opt}| + \log\lceil\frac{1}{\epsilon}\rceil))$. The approximation factor of the previous best known algorithm for the CCPC problem is 1.8841 [25]. Thus, for sufficiently large values of k, our algorithm is much better than the previous one. The hardness result of this problem is unknown, so we can investigate the hardness of the problem as a future work. In future work, we can also consider designing a $(1+\epsilon)$-approximation algorithm for the CCPC problem, where $\epsilon > 0$. Further, we can also consider the following problem as future work: given a convex polygon P and an integer k, the objective is to place k congruent disks centered on the boundary of P such that no two disks intersect, and the radius of disks is maximized.

For the *k-supplier* problem in $I\!R^2$ we have proposed a fixed parameter tractable (FPT) algorithm. Our proposed FPT algorithm produces a 2-approximation result. The running time of the proposed FPT algorithm is $O(6^k(n+m)\log(nm))$, where k is the parameter. We have shown that the proposed FPT algorithm can be easily extended to $I\!R^3$ easily, giving an $(1+\sqrt{2})$-approximation result with running time $O(12^k(n+m)\log(nm))$. Further, we have also shown that the proposed FPT algorithm can be generalized to get an $(1+\epsilon)$-approximation algorithm for the *k-supplier* problem in d-dimensional Euclidean space, where $\epsilon > 0$ is a constant. The running time of the proposed FPT $(1+\epsilon)$-approximation algorithm is $O(\epsilon^{-dk}(n+m)\log(nm))$, where d is a positive integer. For the *k-supplier* problem in $I\!R^2$ we have also proposed a heuristic algorithm based on the Voronoi diagram. An experimental study on our heuristic algorithm was compared with the best known result available in the literature for the *k-supplier* problem in $I\!R^2$ [69]. Experimental results indicate that the proposed heuristic algorithm performs much better than the best known algorithm available in literature [69] with a minor degradation in the running time. Feder and Greene [36] showed that it is NP-hard to approximate the *Euclidean k-supplier* problem less than a factor of $\sqrt{7} \approx 2.64$. Nagarajan et al. [69] gave a 2.74-approximation algorithm for the *Euclidean k-supplier* problem. Since there is a gap between the approximation factor (2.74) of the best known algorithm and the lower bound (2.64), there is still room for further research on the Euclidean k-supplier problem.

Bibliography

[1] Manuel Abellanas, Sergey Bereg, Ferran Hurtado, Alfredo García Olaverri, David Rappaport and Javier Tejel. Moving coins. *Computational Geometry*, Vol. 34(1), pp. 35–48, 2006.

[2] Manuel Abellanas, Natalia de Castro, Gregorio Hernández, Alberto Márquez and Moreno-Jiménez. Gear System Graphs. *Manuscript*, 2006.

[3] Nieves Atienza, Natalia de Castro, Carmen Cortés, M. Ángeles Garrido, Clara I. Grima, Gregorio Hernández, Alberto Márquez, Auxiliadora Moreno-Golzález, Martin Nöllenburg, José Ramón Portillo, Pedro Reyes, Jesús Valenzuela, Maria Trinidad Villar, Alexander Wolff. Cover contact graphs. *Journal of Computational Geometry*, Vol. 3(1), pp. 102–131, 2012.

[4] Pankaj K. Agarwal and Cecilia M. Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, Vol. 33(2), pp. 201–226, 2002.

[5] Pankaj K. Agarwal and Micha Sharir. Efficient algorithm for geometric optimization. *ACM Computing Surveys (CSUR)*, Vol. 30(4), pp. 412–458, 1998.

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

[7] Mark de Berg, Otfried Cheong, Marc Van Kreveld and Mark Overmars. Computational Geometry: Algorithms and Applications. *Springer, Berlin*, 2008.

[8] Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete and Computational Geometry*, Vol. 14(1), pp. 463–479, 1995.

[9] Ahmad Biniaz, Paul Liu, Anil Maheshwari and Michiel Smid. A faster 4-approximation algorithm for the unit disk cover problem. *In Proceedings of 27th Canadian Conference on Computational Geometry (CCCG), Kingston, Ontario, Canada*, 2015.

[10] Prosenjit Bose, Anil Maheshwari, Pat Morin and Jason Morrison. The grid placement problem. *In Proceedings of 7th International Workshop on Algorithms and Data Structures (WADS), Providence, Rhode Island, USA*, LNCS 2125, pp. 180–191, 2001.

[11] Peter Brass, Christian Knauer, Hyeon-Suk Na and Chan-Su Shin. Computing k-centers on a line. *CoRR abs/0902.3282*, 2009.

[12] Prosenjit Bose and Godfried Toussaint. Computing the constrained Euclidean, geodesic and link centre of a simple polygon with applications. *In Proceedings of Computer Graphics International (CGI), Pohang, Korea*, pp. 102–112, 1996.

[13] Adam H. Cannon and Lenore J. Cowen. Approximation algorithms for the class cover problem. *Annals of Mathematics and Artificial Intelligence*, Vol. 40(3-4), pp. 215-224, 2004.

[14] Brent N. Clark, Charles J. Colbourn and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, Vol. 86(1), pp. 165–177, 1990.

[15] 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*, Vol. 2(1), pp. 77–87, 2010.

[16] Paz Carmi, Gautam K. Das, Ramesh K. Jallu, Subhas C. Nandy, Prajwal R. Prasad and Yael Stein. Minimum dominating set problem for unit disks revisited. *Interna-*

tional Journal of Computational Geometry and Applications, Vol. 25(3), pp. 227–244, 2015.

[17] Timothy M. Chan and Nan Hu. Geometric red-blue set cover for unit squares and related problems. *Computational Geometry*, Vol. 48(5), pp. 380–385, 2015.

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

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

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

[21] 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 and Applications*, Vol. 23(6), pp. 461–477, 2013.

[22] Gautam K. Das, Sandip Das and Subhas C. Nandy. Homogeneous 2-hop broadcast in 2D. *Computational Geometry: Theory and Applications*, Vol. 43(2), pp. 182–190, 2010.

[23] Gautam K. Das, Robert Fraser, Alejandro López-Ortiz and Bradford G. Nickerson. On the discrete unit disk cover problem. *International Journal of Computational Geometry and Applications*, Vol. 22(5), pp. 407–420, 2012.

[24] Gautam K. Das, Sasanka Roy, Sandip Das and Subhas C. Nandy. Variations of base station placement problem on the boundary of a convex region. *International Journal of Foundations of Computer Science*, Vol. 19(2), pp. 405–427, 2008.

[25] Hai Du and Yinfeng Xu. An approximation algorithm for k-center problem on a convex polygon. *Journal of Combinatorial Optimization*, Vol. 27(3), pp. 504–518, 2014.

[26] Adrian Dumitrescu and Minghui Jiang. Constrained k-center and movement to independence. *Discrete Applied Mathematics*, Vol. 159(8), pp. 859–865, 2011.

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

[28] Evangelia Pyrga and Saurabh Ray. New existence proofs ϵ-nets. *In Proceedings of the twenty-fourth annual symposium on Computational geometry (SOCG), College Park, MD, USA*, ACM, pp. 199–207, 2008.

[29] Bin Fu, Zhixiang Chen and Mahdi Abdelguerfi. An almost linear time 2.8334-approximation algorithm for the disc covering problem. *In Proceedings of 3rd International Conference on Algorithmic Aspects in Information and Management (AAIM), Portland, Oregon, USA*, LNCS 4508, pp. 317–326, 2007.

[30] Massimo Franceschetti, Matthew Cook and Jehoshua Bruck. A geometric theorem for approximate disk covering algorithms. *Technical Report*, 2001.

[31] Guilherme D. da Fonseca, Celina M. H. de Figueiredo, Vinıcius G. P. de Sá and Raphael Machado. Linear-time sub-5 approximation for dominating sets in unit disk graphs, *In Proceedings of 10th Workshop on Approximation and Online Algorithms (WAOA), Ljubljana, Slovenia*, LNCS 7846, pp. 82–92, 2012.

[32] Stefan Funke, Alex Kesseelman, Fabian Kuhn, Zvi Lotker and Michael Segal. Improved approximation algorithms for connected sensor cover. *Wireless networks*, Vol. 13(2), pp. 153–164, 2007.

[33] Robert Fraser and Alejandro López-Ortiz. The within-strip discrete unit disk cover problem. *In Proceedings of 24th Canadian Conference on Computational Geometry (CCCG), Charlottetown, Prince Edward Island, Canada*, pp. 61–66, 2012.

[34] Robert J. Fowler, Michael S. Paterson and Steven L. Tanimato. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, Vol. 12(3), pp. 133–137, 1981.

[35] Robert J. Fowler, Michael S. Paterson and Steven L. Tanimato. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, Vol. 12(3), pp. 133–137, 1981.

[36] Tomás Feder and Daniel Greene. Optimal algorithms for approximate clustering. *In Proceedings of the twentieth annual ACM symposium on Theory of computing (STOC), Chicago, Illinois, USA*, pp. 434–444, 1988.

[37] Rajiv Gandhi, Samir Khuller and Aravind Srinivasan. Approximation algorithms for partial covering problems. *In Proceedings of the 28th International Colloquium on Automata, Languages and Programming (ICALP)*, LNCS 2076, pp. 225–236, 2001.

[38] Michael R. Garey and David S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. *W.H. Freeman and Company, New York*, 1979.

[39] Hossein Ghasemalizadeh and Mohammadreza Razzazi. An improved approximation algorithm for the most points covering problem. *Theory of Computing Systems*, Vol. 50(3), pp. 545–558, 2012.

[40] Teofilo F. Gonzalez. Covering a set of points in multidimensional space. *Information Processing Letters*, Vol. 40(4), pp. 181–188, 1991.

[41] 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 Algorithms (ESA), Liverpool, United Kingdom*, LNCS 6347, pp. 243–254, 2010.

[42] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, Vol. 38, pp. 293–306, 1985.

[43] Alan Gibbons. Algorithmic Graph Theory. *Cambridge University Press, Cambridge*, 1985.

[44] Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, Vol. 10(2), pp. 180–184, 1985.

[45] Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM (JACM)*, Vol. 33(3), pp. 533–550, 1986.

[46] Dorit S. Hochbaum. Approximation Algorithms for NP-hard Problems. *PWS Publishing Company, Boston*, 1995.

[47] Ferran Hurtado, Vera Sacristán and Godfried Toussaint. Facility location problems with constraints. *Studies in Locational Analysis*, Vol. 15, pp. 17–35, 2000.

[48] 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*, Vol. 18(2), pp. 179–194, 2008.

[49] R. Z. Hwang, Ruei-Chuan Chang and Richard C. T. Lee. The generalized searching over separators strategy to solve some NP-hard problems in subexponential time. *Algorithmica*, Vol. 9(4), pp. 398–423, 1993.

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

[51] Dan Halperin, Micha Sharir and Ken Goldberg. The 2-center problem with obstacles. *Journal of Algorithms*, Vol. 42(1), pp. 109–134, 2002.

[52] Md Iqbal Hossain, Shaheena Sultana, Nazmun Nessa Moon, Tahsina Hashem and Md Saidur Rahman. On Triangle Cover Contact Graphs. *In Proceedings of 9th International Workshop on Algorithms and Computation (WALCOM), Dhaka, Bangladesh*, LNCS 8973, pp. 323–328, 2015.

[53] Takehiro Ito, Shin-Ichi Nakano, Yoshio Okamoto, Yota Otachi, Ryuhei Uehara, Takeaki Uno and Yushi Uno. A polynomial-time approximation scheme for the geo-

metric unique coverage problem on unit squares. *Computational Geometry*, Vol. 51, pp. 25–39, 2016.

[54] János Pach and Pankaj K. Agarwal. Combinatorial geometry. *John Wiley and Sons, New York, NY*, Vol. 37, 1995.

[55] Jiří Matoušek, Raimund Seidel and Emo Welzl. How to net a lot with little: small ϵ-nets for disks and halfspaces. *In Proceedings of the sixth annual symposium on Computational geometry (SOCG), Berkley, CA, USA*, ACM, pp. 16–22, 1990.

[56] Jiří Matoušek. Lectures in Discrete Geometry. *Springer-Verlag, New York, NY*, Vol. 108, 2002.

[57] Haim Kaplan, Matthew J. Katz, Gila Morgenstern and Micha Sharir. Optimal cover of points by disks in a simple polygon. *SIAM Journal on Computing*, Vol. 40(6), pp. 1647–1661, 2011.

[58] Matthew J. Katz and Gila Morgenstern. A scheme for computing minimum covers within simple regions. *Algorithmica*, Vol. 62(1-2), pp. 349–360, 2012.

[59] Arindam Karmakar, Sandip Das, Subhas C. Nandy and Binay K. Bhattacharya. Some variations on constrained minimum enclosing circle problem. *Journal of Combinatorial Optimization*, Vol. 25(2), pp. 176–190, 2013.

[60] Kenneth L. Clarkson and Kasturi Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete and Computational Geometry*, Vol. 37(1), pp. 43–58, 2007.

[61] Sung Kwon Kim and Chan-Su Shin. Efficient algorithm for two-center problems for a convex polygon. *In Proceedings of 6th Annual International Computing and Combinatorics Conference (COCOON), Sydney, Australia*, LNCS 1858, pp. 299–309, 2000.

[62] Erik Jan van Leeuwen. Optimization and approximation on systems of geometric objects. *Ph.D. Thesis, University of Amsterdam*, 2009.

[63] Paul Liu and Daniel Lu. A fast $\frac{25}{6}$-approximation algorithm for the minimum unit disk cover problem. *CoRR abs/1406.3838*, 2014.

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

[65] Priya Ranjan Sinha Mahapatra, Partha P. Goswami and Sandip Das. Covering points by isothetic unit squares. *In Proceedings of 19th Canadian Conference on Computational Geometry (CCCG), Ottawa, Canada*, pp. 169–172, 2007.

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

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

[68] Nabil Hassan Mustafa and Saurabh Ray. PTAS for geometric hitting set problems via local search. *In Proceedings of the twenty-fifth annual symposium on Computational geometry (SOCG), arhus, Denmark*. ACM, pp. 17-22, 2009.

[69] Viswanath Nagarajan, Baruch Schieber and Hadas Shachnai. The Euclidean k-Supplier Problem. *In Proceedings of the 16th International Conference on Integer Programming and Combinatorial Optimization (IPCO), Valparaso, Chile*, LNCS 7801, pp. 290–301, 2013.

[70] Sada Narayanappa and Petr Vojtechovskỳ. An improved approximation factor for the unit disk covering problem. *In Proceedings of 18th Canadian Conference on Computational Geometry (CCCG), Kingston, Ontario, Canada*, pp. 15–18, 2006.

[71] Sasanka Roy, Debabrata Bardhan, Sandip Das. Base station placement on boundary of a convex polygon. *Journal of Parallel and Distributed Computing*, Vol. 68(2), pp. 265–273, 2008.

[72] 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 twenty-ninth annual ACM symposium on Theory of computing (STOC), El Paso, Texas, USA*, pp. 475–484, 1997.

[73] Atsuo Suzuki and Zvi Drezner. The p-center location problem in area. *Location Science*, Vol. 4(1), pp. 69–82, 1996.

[74] Chandan Saha and Sandip Das. Covering a set of points in a plane using two parallel rectangles. *Information Processing Letters*, Vol. 109(16), pp. 907–912, 2009.

[75] Min-Te Sun and Ten-Hwang Lai. Computing optimal local cover set for broadcast in ad hoc networks. *In Proceedings of the IEEE International Conference on communication (ICC), New York, NY, USA*, Vol. 5, pp. 3291–3295, 2002.

[76] Min-Te Sun, Xiaoli Ma, Chih-Wei Yi, Chuan-Kai Yang and Ten H. Lai. Minimum disc cover set construction in mobile ad hoc networks. *In Proceedings of the 3rd International Conference on Computer Network and Mobile Computing (ICCNMC), Zhangjiajie, China*, LNCS 3619, pp. 712–721, 2005.

[77] Min-Te Sun, Chih-Wei Yi, Chuan-Kai Yang and Ten-Hwang Lai. An optimal algorithm for the minimum disc cover problem. *Algorithmica*, Vol. 50(1), pp. 58–71, 2008.

[78] Thomas Erlebach and Erik Jan Leeuwen. PTAS for weighted set cover on unit squares. *In Proceedings of 13th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX), and 14th International Workshop on Randomization and Computation (RANDOM), Barcelona, Spain*, LNCS 6302, pp. 166–177, 2010.

[79] Steven L. Tanimoto and Robert J. Fowler. Covering image subsets with patches. *In Proceedings of the 5th International Conference on Pattern Recognition, Miami Beach, Florida, USA*, Vol. 2, pp. 835–839, 1980.

[80] Dejun Yang, Satyajayant Misra, Xi Fang, Guoliang Xue and Junshan Zhang. Two-tiered constrained relay node placement in wireless sensor networks: computational

complexity and efficient approximations. *IEEE Transactions on Mobile Computing*, Vol. 11(8), pp. 1399–1411, 2012.

[81] 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*, Vol. 412(3), pp. 198–208, 2011.

Publication from the Contents of the Thesis

Papers published/submitted in International Journals:

[J1] Manjanna Basappa, Rashmisnata Acharyya and Gautam K. Das, "Unit Disk Cover Problem in 2D", *Journal of Discrete Algorithms (Elsevier)*, Vol. 33, pp. 193-201, 2015.

[J2] Manjanna Basappa, Ramesh K. Jallu and Gautam K. Das, "Constrained k-Center Problem on a Convex Polygon", *International Journal of Foundations of Computer Science (World Scientific)*(Submitted).

[J3] Manjanna Basappa and Gautam K. Das, "Discrete Unit Square Cover Problem", *Discrete Mathematics, Algorithms and Applications (World Scientific)*(Submitted).

Papers Published in International Conference Proceedings:

[C1] Manjanna Basappa, Ramesh K. Jallu, Gautam K. Das and Subhas C. Nandy, "The Euclidean k-Supplier Problem in $\mathbb{R}^2$" In *Proceedings of 12th International Symposium on Algorithms and Experiments for Wireless Sensor Networks (ALGOSENSORS 2016)*, (Springer-Verlag), Aarhus, Denmark, August 25-26, 2016 (Accepted).

[C2] Manjanna Basappa, Ramesh K. Jallu and Gautam K. Das, "Constrained k-Center Problem on a Convex Polygon" In *Proceedings of 15th International Conference on Computational Science and its Applications (ICCSA 2015)*, LNCS 9156 (Springer-Verlag), Banff, Alberta, Canada, pp. 209-222, June 22-25, 2015.

[C3] Rashmisnata Acharyya, Manjanna Basappa and Gautam K. Das, "Unit Disk Cover Problem in 2D" In *Proceedings of 13th International Conference on Computational Science and its Applications (ICCSA 2013)*, LNCS 7972 (Springer-Verlag), Ho Chi Minh City, Vietnam, pp. 73-85, June 24-27, 2013.