Algorithms for Facility Location Problems in Geometric Settings

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

of

Doctor of Philosophy

in

Computer Science and Engineering

Submitted by
Pawan Kumar Mishra

Under the guidance of
Prof. S. V. Rao
Prof. Gautam Kumar Das

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

Dedicated to

My Family

Abstract

Facility Location Problems (FLPs) have been the subject of extensive research because of its diverse range of applications in VLSI, networks, clustering, and other areas. The *covering* problem and the *dispersion* problem are two popular FLPs. Covering problem refers to selecting a subset of covering objects from a given set of objects such that the union of the selected objects contains all the elements. On the other hand, the dispersion problem refers to selecting a subset of a given set of objects such that the closeness between the objects in the selected set is undesirable. In this thesis, we investigate covering and dispersion problems in geometric settings.

One of the most studied covering problems is the *set cover* (SC) problem, where the objective is to find a minimum number of objects such that the union of all the selected objects covers all the elements. We introduce a natural generalization of the SC problem, where each covering object has a bound (capacity) on the number of elements it can cover. The problem is referred to as the *capacitated set cover* (CSC) problem. We study the CSC problem in geometric settings, where a set of unit disks is considered as a set of covering objects, and the capacity of each unit disk is uniform. The problem is called the (α, P, Q)-covering problem, and it is defined as follows.

For a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points and a set $Q = \{q_1, q_2, \ldots, q_m\}$ of m points and a positive integer α, a subset $Q' \subseteq Q$ is said to be an α-cover of P if (i) the point set P can be partitioned into ℓ subsets $P_1, P_2, \ldots, P_\ell$ such that $|P_i| \leq \alpha$ for each $i = 1, 2, \ldots \ell$, where $\ell = |Q'|$ and (ii) each $p \in P_i$ is covered by a unit disk centered at $q'_i \in Q'$. Given a positive integer α, a set P of n points and a set Q of m points, the objective of the (α, P, Q)-covering problem is to find a minimum cardinality α-cover $Q' \subseteq Q$ of P.

We establish a necessary and sufficient condition through which one can ensure whether

the given instance is feasible or not. Further, we prove that the (α, P, Q)-covering problem is NP-complete for $\alpha \geq 3$. Finally, we propose a local search algorithm for the (α, P, Q)-covering problem, and prove that the proposed algorithm is a PTAS.

For the dispersion problem, we introduce the concept of dispersion partial sum, which generalizes the notion of dispersion. Based on the dispersion partial sum, we define variants of the dispersion problem, namely the 1-dispersion problem, the 2-dispersion problem, and the c-dispersion problem. We study the following dispersion problems in Euclidean space: the 2-dispersion problem in both $\mathbb{R}^1$ and $\mathbb{R}^2$ and the 1-dispersion problem in $\mathbb{R}^2$. The 2-dispersion problem is defined as follows.

Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points, the non-negative distances between each pair of points $p_i, p_j \in P$, and a positive integer k $(3 \leq k \leq n)$, for each point $p \in P$ and a subset S of P, the 2-dispersion cost of the point $p \in P$ with respect to S, $cost_2(p, S)$, is defined as the sum of Euclidean distances from p to the closest point in $S \setminus \{p\}$ and the second closest point in $S \setminus \{p\}$. The 2-dispersion cost of the subset S is defined as $cost_2(S) = \min_{p \in S}\{cost_2(p, S)\}$. The objective of the 2-dispersion problem is to find a subset $S \subseteq P$ of cardinality k such that $cost_2(S)$ is maximized.

In this thesis, we present a $(2\sqrt{3} + \epsilon)$-factor approximation result for the 2-dispersion problem in $\mathbb{R}^2$. We also develop a common framework for the dispersion problem in Euclidean space, which produces a $2\sqrt{3}$-factor approximation result and an optimal result for the 2-dispersion problem in $\mathbb{R}^2$ and $\mathbb{R}^1$, respectively.

Next, we study the 1-dispersion problem, which is defined as follows.

Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points, the non-negative distances between each pair of points $p_i, p_j \in P$, and a positive integer k $(2 \leq k \leq n)$, for each point $p \in P$ and a subset S of P, the 1-dispersion cost of the point $p \in P$ with respect to S, $cost_1(p, S)$, is defined as a distance of p to the closest point in $S \setminus \{p\}$. The 1-dispersion cost of the

subset $S \subseteq P$ *is defined as* $cost_1(S) = \min_{p \in S}\{cost_1(p, S)\}$. *The objective of the* 1*-dispersion problem is to find a subset* $S \subseteq P$ *of cardinality* k *such that* $cost_1(S)$ *is maximized.*

In this thesis, we propose a 2-factor approximation result for the 1-dispersion problem in $\mathbb{R}^2$.

Next, we study the dispersion problem in a metric space. We introduce a variant of the dispersion problem, namely the c-dispersion problem. The problem is defined formally as follows.

Given a set $P = \{p_1, p_2, \ldots, p_n\}$ *of* n *points, the non-negative distance* $d(p_i, p_j)$ *between each pair of points* $p_i, p_j \in P$, *and a positive integer* k $(c+1 \leq k \leq n)$, *for each point* $p \in P$ *and a subset* S *of* P, *the* c*-dispersion cost of the point* $p \in P$ *with respect to* S, $cost_c(p, S)$, *is defined as the sum of distances from* p *to the* c *closest points in* $S \setminus \{p\}$. *The* c*-dispersion cost of the subset* S *of* P *is defined as* $cost_c(S) = \min_{p \in S}\{cost_c(p, S)\}$. *The objective of the* c*-dispersion problem is to find a* k *size subset* S *of* P *such that* $cost_c(S)$ *is maximized.*

In this thesis, we propose a greedy algorithm for the c-dispersion problem, which produces a $2c$-factor approximation result. We also prove that the c-dispersion problem in a metric space parameterized by solution size is W[1]-hard.

Finally, we consider a variant of the 1-dispersion problem, where a set of locations is the vertices of a convex polygon. This variant of the 1-dispersion problem is referred to as the *convex* 1*-dispersion* problem, and it is defined as follows.

Given a set $P = \{p_1, p_2, \ldots, p_n\}$ *of* n *vertices of a convex polygon, the Euclidean distance* $d(p, q)$ *between each pair of vertices* $p, q \in P$, *the objective of the convex* 1*-dispersion problem is to find a subset* S *of* P *of size* k *such that the cost of a subset* S, $cost(S) = \min\{d(p, q) \mid p, q \in S\}$, *is maximized.*

In this thesis, we propose an $O(n^3)$-time algorithm that returns an optimal result where the objective is to select $k(= 4)$ vertices for the convex 1-dispersion problem. We also

propose a $\sqrt{3}$ (≈ 1.733)-factor approximation algorithm for the convex 1-dispersion problem for any value of k.

Declaration

I certify that:

- The work contained in this thesis is original and has been done by myself and under the general supervision of my supervisors.
- The work reported herein has not been submitted to any other Institute for any degree or diploma.
- Whenever I have used materials (concepts, ideas, text, expressions, data, graphs, diagrams, theoretical analysis, results, etc.) from other sources, I have given due credit by citing them in the text of the thesis and giving their details in the references. Elaborate sentences used verbatim from published work have been clearly identified and quoted.
- I also affirm that no part of this thesis can be considered plagiarism to the best of my knowledge and understanding and take complete responsibility if any complaint arises.
- I am fully aware that my thesis supervisors are not in a position to check for any possible instance of plagiarism within this submitted work.

Date:

Place: Guwahati

(Pawan Kumar Mishra)

Acknowledgements

The pursuit of a PhD is a lengthy endeavor, and it would not have been possible to reach this destination without the gracious support of a number of individuals. I would like to take this opportunity to express my sincere gratitude to them for the kind help they have provided.

In the beginning, I thank my supervisors, Professor S.V. Rao and Professor Gautam K. Das, for all the guidance, encouragement, and assistance they have provided to me over the last five years. They have allowed me to investigate new directions within the scope of the research, which I greatly appreciate. I owe them a lot because they always pointed out my simple mistakes and kept me interested in my Ph.D. work, and I thank them very much for that.

I express my gratitude to Professor G. Sajith, Dr. Pinaki Mitra, and Dr. Deepanjan Kesh, members of my doctorate committee, for the insightful remarks and ideas they provided throughout the course of my Ph.D. In addition, I express my gratitude to the department technical officers, technical superintendents, and administrative personnel for the complete and unconditional support they have provided.

My younger brother, Pankaj, was always there for me when I needed him the most and I would like to express my gratitude to him. He solved all the problems (not related to research problems, though ;-)), making the transition to my Ph.D. much less stressful.

My parents have shown support, patience, love, blessings, and unwavering faith. For all of this, I am sincerely grateful and indebted to all of them. Also, obtaining a Ph.D. while staying away from home is a challenging task unless there is someone who can look after the parents. Ruby, my sister, is the person to whom I owe a debt and for whom I will be grateful until the end of time.

My girlfriend, Kasturi, has been incredibly supportive and caring and I am beyond grateful for all she has done for me. She has faith in my abilities and has always supported me.

In addition, I would like to thank two toddlers, Chuha and TukTuk, who were stress busters during the last phase of my Ph.D.

I thank all my friends, Akshay, Alakesh, Arijit, Aurobindo, Bala Bhaiya, Bhale, Chaitanya, Dip, Gyanendra, Harish Da, Hemanta, Jenil, Kamal, Mirza Bhai, Naro, Nayan, Omesh, Palash Da, Pallav, Pankaj, Parikshit Da, Pritam, Sangram, Ujjwal Da, with whom I have spent most of the time.

Last but not the least, I would like to thank those people who discouraged, demoralized, and menaced me at different stages of life. All of this motivated me to take something as a challenge and work hard for it.

Certificate

This is to certify that this thesis entitled, **"Algorithms for Facility Location Problems in Geometric Settings"**, being submitted by **Pawan Kumar Mishra**, to the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, for partial fulfillment of the award of the degree of Doctor of Philosophy, is a bonafide work carried out by him under my supervision and guidance. The thesis, in my opinion, is worthy of consideration for award of the degree of Doctor of Philosophy in accordance with the regulation of the institute. To the best of my knowledge, it has not been submitted elsewhere for the award of the degree.

Date:

Place: Guwahati

.....

Prof. S. V. Rao

Professor

Department of Computer Science and Engineering

IIT Guwahati

.....

Prof. Gautam Kumar Das

Professor

Department of Mathematics

IIT Guwahati

Contents

Abstract iv

Declaration viii

Acknowledgement ix

Certificate xi

List of Figures xv

List of Algorithms xviii

List of Abbreviations xix

List of Symbols xx

1 Introduction 1

1.1 Preliminaries 13

1.2 Scope of the Thesis 15

1.3 Organization of the Thesis 17

2 Literature Review **20**

2.1 Covering Problem 21

2.2 Dispersion Problem 24

3 Capacitated Discrete Unit Disk Cover Problem **29**

3.0.1 Overview of the Chapter 30

3.1 A Necessary and Sufficient Condition 30

3.2 Hardness of the (α, P, Q)-Covering Problem 33

3.3 A PTAS 42

3.4 Conclusion 47

4 Euclidean Dispersion Problems **49**

4.0.1 Overview of the Chapter 50

4.1 $(2\sqrt{3}+\epsilon)$-Factor Approximation Algorithm 51

4.2 A Common Framework for the Euclidean Dispersion Problem 57

4.2.1 $2\sqrt{3}$-Factor Approximation Result for the 2-Dispersion Problem in $\mathbb{R}^2$ 58

4.2.2 2-Dispersion Problem on a Line 64

4.2.3 1-Dispersion Problem in $\mathbb{R}^2$ 68

4.3 Conclusion 72

5 Dispersion Problem in a Metric Space **74**

5.0.1 Overview of the Chapter 75

5.1 Algorithm 75

5.2 A Parameterized Reduction 84

5.3 Conclusion 86

6 Convex 1-Dispersion Problem **88**

6.0.1 Overview of the Chapter 89

6.1 Convex 1-Dispersion Problem for $k = 4$ 89

6.1.1 Preliminaries 90

6.1.2 Algorithm 92

6.2 $\sqrt{3}$-Factor Approximation Result for the Convex 1-Dispersion Problem . . . 100

6.3 Conclusion 105

7 Conclusion and Future Work **107**

References **111**

List of Figures

1.1 Unit disks as a set of geometric objects 5

1.2 (a) An instance of the (α, P, Q)-covering problem, (b) Unit disk centered on the points of Q, (c) Optimal solution when $\alpha = 3$, and (d) Optimal solution when $\alpha = 2$ 6

1.3 Dispersion sum of facility placed at location p_1 9

1.4 (a) dispersion sum metric, and (b) dispersion partial sum metric 11

1.5 Convex Polygon 13

3.1 (a) An instance of the (α, P, Q)-covering problem, and (b) Construction of bipartite graph for $\alpha = 2$, here vertex v_{ij} represent j^{th} copy of disk i. 32

3.2 (a) A planar graph G of maximum degree 3, and (b) p_i for each $v_i \in V$. . . . 35

3.3 (a) placement of joint points where $\ell' = 1$, (b) Placement of joint points where $\ell' > 1$, and (c) placement of support points with respect to p_1. 36

3.4 (a) Added points and line segments in the embedding, and (b) instance of the $(3, P, P)$-covering problem. 38

3.5 (a) A vertex cover $\{v_1, v_3, v_4\}$ of G, and (b) the construction of J' and S'. . . 40

4.1 Points $p_a, p_b, p_c \in D[p_i]$ . . . 52
4.2 $H(D[OPT'] \cup S_i, \mathcal{E})$ . . . 53
4.3 2-dispersion cost of p_t^* with respect to OPT. . . . 54
4.4 p_ℓ lies outside the disk $D[p_j^*]$ . . . 56
4.5 q is not contained in $D[p_j^*]$ . . . 56
4.6 2-dispersion cost of p_m^* with respect to S^* . . . 61
4.7 p_ℓ lies outside of the disk $D[p_j^*]$ . . . 63
4.8 q is not contained in $D[p_j^*]$ . . . 64
4.9 s_r^* and s_t^* on left side of s_o^* . . . 65
4.10 Snippet of S_4' . . . 66
4.11 Placement of set $S_{i-1} \cup \{p_i\}$. . . . 67
4.12 Closest point of p_j^* lies outside of $D(p_j^*)$. . . . 71

5.1 $q \in B(p)$, but $r \notin B(p)$. . . . 77
5.2 Illustration of Lemma 5.1.1. . . . 77
5.3 $G(B(\overline{X}) \cup M, \mathcal{E})$. . . . 79
5.4 c-dispersion cost of p_t^* with respect to S^* . . . 80
5.5 $p \in B(p_j^*)$, and $q \notin B(p_j^*)$. . . . 83

6.1 (a) L_{ij}, R_{ij} for the pair $(p_i, p_j) \in P$, and (b) $R_{ij} = \phi$ . . . 90
6.2 S_{ij} does not exist for the pair (p_i, p_j) . . . 91
6.3 (a) p_k and p_ℓ are in L_{ij}, (b) p_k and p_ℓ are in R_{ij}, and (c) p_k is in L_{ij} and p_ℓ is in R_{ij} . . . 91
6.4 Illustration of Observation 6.1.1 . . . 95
6.5 Illustration of Lemma 6.1.2 . . . 96
6.6 Illustration of Base Case . . . 102

6.7 Illustration of Inductive Step 103

List of Algorithms

1 Local_Search_Algorithm(α, P, Q) 43

2 Euclidean_Dispersion_Algorithm(P, k) 52

3 Framework_Euclidean_Dispersion(P, k, γ) 58

4 Metric_Dispersion_Algorithm(P, k) 76

5 Convex 1-Dispersion_Algorithm$(P, 4)$ 94

6 Convex 1-Dispersion_Algorithm(P, k) 101

List of Abbreviations

NP	The class of non-deterministic polynomial-time solvable problems
NP-hard	The class of non-deterministic polynomial-time hard problems
OPT	Optimal solution
P	The class of deterministic polynomial-time solvable problems
PTAS	Polynomial-time approximation scheme
FPTAS	Fully polynomial-time approximation scheme
VLSI	Very large scale integration
SDN	Software Defined Networking
FLPs	Facility Location Problems
SC	Set Cover
CSC	Capacitated Set Cover
VC	Vertex Cover
GSC	Geometric Set Cover
GCSC	Capacitated Geometric Set Cover
DUDC	Discrete Unit Disk Cover
IS	Independent set

List of Symbols

Symbol	Meaning
$\lfloor x \rfloor$	The largest integer less than or equal to x
$\lceil x \rceil$	The smallest integer greater than or equal to x
$a \leftarrow b$	Variable a gets the value of b
$\sum$	The addition of a sequence of numbers
$\forall$	For all
$\exists$	There exists
$T(n)$	The time complexity of an algorithm with input size n
$O(\cdot)$	The asymptotic big-oh notation
$\mid$	Such that
$\min\{x, y\}$	The minimum of x and y
$\square$	The end of a proof
$\emptyset$	The empty set
$\{.\}$	The set notation
$\mathbb{R}^2$	$\{(a, b) \mid a, b \in \mathbb{R}\}$
$\mathcal{P}$	A set of points in $\mathbb{R}^2$
$p \in \mathcal{P}$	p is a member of $\mathcal{P}$
$d(p, q)$	The Euclidean distance between p and q
$\mathcal{Q} \subseteq \mathcal{P}$	$\mathcal{Q}$ is a subset of $\mathcal{P}$

$\mathcal{P} \cap \mathcal{Q}$	The intersection of $\mathcal{P}$ and $\mathcal{Q}$
$\mathcal{P} \cup \mathcal{Q}$	The union of $\mathcal{P}$ and $\mathcal{Q}$
$\mathcal{P} \setminus \mathcal{Q}$	The set minus of $\mathcal{P}$ and $\mathcal{Q}$
$\lvert \cdot \rvert$	The cardinality of a set
k	A positive integer
$N[p]$	The closed neighborhood of a point p
$N(p)$	The open neighborhood of a point p
$D(p)$	The open unit disk centered at a point p
$D(P)$	The set of open unit disks centered at the points in P
$D[p]$	The closed unit disk centered at a point p
$D[P]$	The set of closed unit disks centered at the points in P
$B(p)$	The open unit ball centered at a point p
$B(P)$	The set of open unit balls centered at the points in P
S_i	A subset containing i points having certain properties
$G = (V, E)$	An undirected connected simple graph with vertex set V and edge set E
$e \in E$	e is a member of E
n	The cardinality of V (or) the cardinality of a point set $\mathcal{P}$
m	The cardinality of E
$N_G(.)$	The open neighborhood of a vertex/set
$\{\mathcal{A}_\epsilon\}$	A collection of algorithms with ϵ as input parameter

1

Introduction

Facility location problems (FLPs) have been at the core of optimization problems in the twentieth century and have thus been extensively studied [33,34,39,40]. In FLP, the fundamental challenge is to determine where to locate the facilities to achieve specific objectives. In general, facilities are warehouses, franchises, hospitals, oil tanks, *etc.* [62,74]. However, the objectives are mainly a function of the cost of installing the facilities or a function of the distances between the installed facilities [33,34]. Due to the wide range of applications in the real world, FLPs continue to be a topic of extensive and ongoing research interest. Some

of the popular FLP applications include VLSI design, image processing, and clustering. In this thesis, we study two popular FLPs in geometric settings: the *covering* problem, and the *dispersion* problem. The covering problem refers to selecting a subfamily of sets from a given family of sets such that the union of the selected sets contains all the elements of the specified ground set. On the other hand, the dispersion problem refers to finding a subset of a given set of elements such that proximity (closeness) between the elements in the selected set is undesirable. Note that the objective of the covering problem is a function of the cost of the installation of the facilities. In contrast, the objective of the dispersion problem is a function of the distance between the installed facilities. In the literature, the dispersion problem is closely related to another popular FLP, namely the *Anti-Covering* problem[1].

When it comes to applications of covering and dispersion problems, the possibilities are enormous. However, only a few of them are included here. The covering problem has many applications in various domains, including wireless networks [1, 16, 53], VLSI [56], image processing [19], computational biology [57], and computational learning theory [59, 60]. Similarly, the dispersion problem has many applications, including the placement of hazardous structures, the opening of franchise stores [37], geographic analysis [63], and information retrieval [11, 50]. See more applications of the dispersion problem in [18, 64, 70, 75].

In the covering problem, a set system $(\mathcal{U}, \mathscr{S})$ is given, where $\mathcal{U}$ is the set consisting of elements and $\mathscr{S}$ is a family of subsets of $\mathcal{U}$. The objective is to find a subfamily $\mathcal{F} \subseteq \mathscr{S}$ such that its union contains each element in $\mathcal{U}$. The subfamily $\mathcal{F}$ is referred to as *covering set* of $\mathcal{U}$. One can model the covering problem in a number of different ways, depending on the kind of application being considered. To better understand the modeling, we refer

[1] In the Anti-Covering problem, the objective is to select the largest possible subset of elements from a given set of elements such that each element in the selected subset is at least a distance r units away from other elements in the selected subset [66]. This problem is also known as the r-separation problem [38].

to an element in $\mathcal{U}$ as a "client" and a set in $\mathscr{S}$ as a "service provider". Now, if a client $c \in \mathcal{U}$ belongs to a service provider $S \in \mathscr{S}$, we say that S can provide a service to the client c. Similarly, $S \in \mathscr{S}$ contains a set of clients for which it can potentially provide a service. Based on the concept of clients and service providers, we consider a few scenarios that can be modeled using the covering problem. For example, we might need to find the minimum number of service providers to serve all clients. This problem is referred to as the *set cover* (SC) problem. In the SC problem, the objective is to find the minimum cardinality covering set $\mathcal{F} \subseteq \mathscr{S}$. Now consider another application as follows: for each client, there are exactly two service providers that can potentially serve it, and we need to find the minimum number of service providers to serve all clients. This problem is called the *vertex cover* (VC) problem. Furthermore, in some applications, we might have a capacity constraint for each service provider, *i.e.*, a bound on the number of clients it can serve, and considering the capacity constraint, we wish to find the minimum number of service providers to serve all clients. This problem is referred to as the *capacitated set cover* (CSC) problem, and it is also a natural variant of the SC problem. In the CSC problem, in addition to a set system $(\mathcal{U}, \mathscr{S})$, a capacity C_i is given for each set $S_i \in \mathscr{S}$. Here, C_i is a bound on the number of elements that S_i can cover (also referred to as the capacity of the set S_i). Note that the SC problem is a special case of the CSC problem, where the capacity of each set $S_i \in \mathscr{S}$ is unbounded. Based on the number of copies of each set $S_i \in \mathscr{S}$, there are two versions of the CSC problem, namely (i) soft, and (ii) hard capacitated versions. Each set $S_i \in \mathscr{S}$ has an unbounded number of copies available in the soft capacitated version. In contrast, each set $S_i \in \mathscr{S}$ has a bound on the number of copies available in the hard capacitated version. Now consider an application in which the services provided by a service provider are restricted to a certain geographical region. This means that a service provider can only provide its services to clients located within the boundaries of its geographical region. It is quite intuitive to use

a geometric object to represent the geographical boundary of the service provider, while a point can be used to represent clients. A client (point) within the geographical boundary of the service provider (geometric object) means that the service provider can serve the client. Taking into account the above, the SC problem can be defined in geometric settings. In the *geometric set cover* (GSC) problem, a range space, analogous to a set system, is defined as $S = (X, \mathscr{R})$, where X is a set of points (finite or infinite) and $\mathscr{R}$ is a (finite or infinite) family of subsets of X called ranges. Ranges are defined by the intersection of X and geometric objects such as unit disks, unit squares, axis-parallel rectangles, and, in general, convex pseudo-disks. The objective is to find a minimum cardinality $R' \subseteq \mathscr{R}$ of the ranges such that all points in X are covered, *i.e.*, for all $p \in X$, there exists $\mathbf{r} \in R'$ such that $p \cap \mathbf{r} \neq \phi$. The GSC problem is a special case of the general SC problem. Although the general SC problem is NP-hard to approximate within a factor of $(1-\epsilon)\log n$ [31], some GSC problems admit a PTAS due to the geometric properties of the object. Now, if there is a bound on the number of points that a geometric object in $\mathscr{R}$ can cover, *i.e.*, the capacity of each $R_i \in \mathscr{R}$, then the GSC problem is known as the *geometric capacitated set cover* (GCSC) problem. In the GCSC problem, if a set of unit disks are considered as a set of geometric objects (see Figure 1.1) and the capacity of each unit disk is uniform, then the GCSC problem is referred to as the (α, P, Q)-*covering* problem, and it is defined as follows.

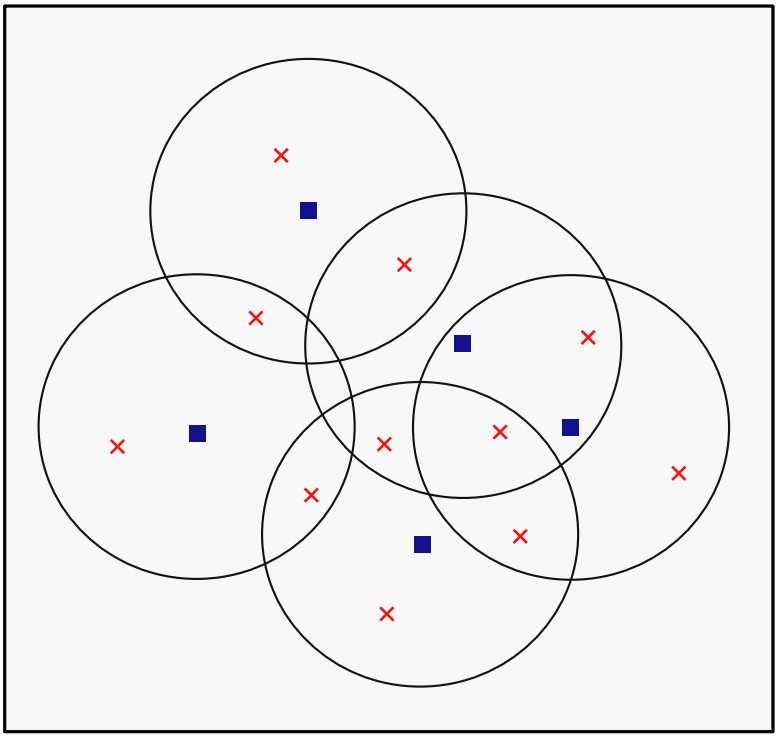


Figure 1.1: *Unit disks as a set of geometric objects*

(α, P, Q)-Covering Problem: *For a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points and a set $Q = \{q_1, q_2, \ldots, q_m\}$ of m points and a positive integer α, a subset $Q' \subseteq Q$ is said to be an α-cover of P if (i) the point set P can be partitioned into ℓ subsets $P_1, P_2, \ldots, P_\ell$ such that $|P_i| \leq \alpha$ for each $i = 1, 2, \ldots \ell$, where $\ell = |Q'|$ and (ii) each $p \in P_i$ is covered by a unit disk centered at $q'_i \in Q'$. Given a positive integer α, a set P of n points and a set Q of m points, the objective of the (α, P, Q)-covering problem is to find a minimum cardinality α-cover $Q' \subseteq Q$ of P.*

In Figure 1.2(c), if the capacity (α) of each disk is 3, then the set $Q' = \{q_1, q_3\}$ is the optimal solution, as the unit disks centered on q_1 and q_3 cover all the points. Similarly, in Figure 1.2(d), if the capacity (α) of each disk is 2, then the set $Q' = \{q_1, q_2, q_3\}$ is the

optimal solution, as the unit disks centered on q_1, q_2 and q_3 cover all the points.

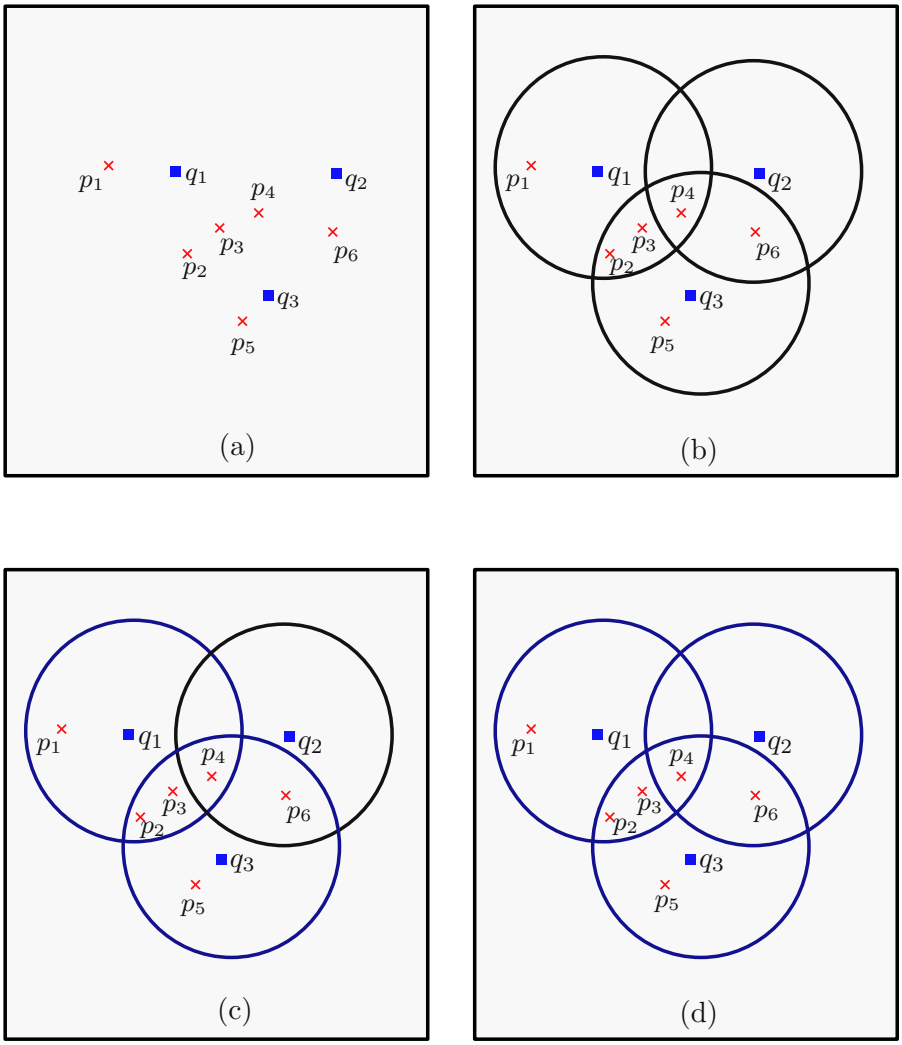


Figure 1.2: *(a) An instance of the (α, P, Q)-covering problem, (b) Unit disk centered on the points of Q, (c) Optimal solution when $\alpha = 3$, and (d) Optimal solution when $\alpha = 2$*

Motivations. Our interest in the (α, P, Q)-covering problem arises from the coverage problem in software-defined networking (SDN). In SDN, the control plane (which controls traffic routing) is decoupled from the data plane (packet forwarding). The *switches* are responsible for the data plane and *controllers* for the control plane. The SDN controller collects information from the switches that fall into the controller's coverage area. Depending on

the price of the installation of a controller, there can be a limitation on the number of switches that a controller can communicate irrespective of the number of switches falling in the controller's range. In the literature, this problem is addressed as a capacitated controller problem in SDN [83]. The constraint on the number of switches controlled by a controller inspired us to study the (α, P, Q)-covering problem.

Next, we consider the dispersion problem. In a typical dispersion problem, a set $P = \{p_1, p_2, \ldots, p_n\}$ of n locations, a positive integer k and a distance function $d(.,.)$ are given. Here, P is a set of desired locations where facilities can be placed. The goal is to select a k size subset S of P and place the facilities in each location in S so that proximity (closeness) between the facilities in S is undesirable. More specifically, in the dispersion problem, the goal is to minimize interference between the facilities located in S. In the literature, there are multiple variants of the dispersion problem; of which the two popular variants of the dispersion problem are as follows: (i) *Max-Min-Min*, and (ii) *Max-Min-Sum*.

In the Max-Min-Min dispersion problem, k facilities are to be placed in order to maximize the minimum distance that separates two facilities. We refer to the Max-Min-Min problem as the 1-*dispersion* problem, and it is defined formally as follows.

1-Dispersion Problem: *Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points, the non-negative distances between each pair of points $p_i, p_j \in P$, and a positive integer k $(2 \leq k \leq n)$, for each point $p \in P$ and a subset S of P, the 1-dispersion cost of the point $p \in P$ with respect to S, $cost_1(p, S)$, is defined as a distance of p to the closest point in $S \setminus \{p\}$. The 1-dispersion cost of the subset $S \subseteq P$ is defined as $cost_1(S) = \min_{p \in S}\{cost_1(p, S)\}$. The objective of the 1-dispersion problem is to find a subset $S \subseteq P$ of cardinality k such that $cost_1(S)$ is maximized.*

Motivations. When we investigate the 1-dispersion problem in the logistics context in the real world, we find a huge variety of applications. These applications include the establishment of franchise stores, the installation of missile launch pads, the establishment of hazardous structures such as nuclear power plants and oil tanks, and many more. Now, we will discuss some of the applications of the 1-dispersion problem. Consider a scenario where there exist n desired locations/sites available to open fast food franchise stores in an area; the objective is to select exactly k ($\leq n$) locations from n desired locations so that stores located in these selected k locations are mutually far from each other. Basically, it is not desirable to have two stores that share the same clientele. Another instance where the problem of dispersion arises is during the installation of missile launch pads. When multiple launch pads are grouped together in the same area, it is possible for a single adversary to eliminate all of them at once. Therefore, these launch pads must be distributed to the greatest extent possible to ensure that an accident on one of the launch pads has no implications for others. In addition to logistic settings, the 1-dispersion problem has various applications, provided that we can define the right measure of closeness. One of such possible applications in which the 1-dispersion problem could be used is in the introduction of new products to the market; however, this would require us to define the appropriate closeness between the products. Suppose that a company wants to introduce k new diverse products from a total of n possible new products and wants to do so in such a way that the products launched are diverse in nature (with regard to quality, price, shape, *etc.*.) Therefore, if an appropriate measure of the closeness between products is defined, then the 1-dispersion problem can be used to find k new diverse products from a total of n possible new products. The 1-dispersion problem also has application in information retrieval, where we need to locate a small subset of data with some desirable variety from an extended data collection such that a small subset may be used as a fair sample to provide an overview of the large data set [11, 50].

Next, to define the Max-Min-Sum problem, a metric *dispersion sum* is introduced. To define the dispersion sum, assume that k locations are selected (to place facilities) from n desired locations (see Figure 1.3(a)). Then, for a facility, the dispersion sum depends on the other $k-1$ located facilities. The dispersion sum of a facility is the sum of the distances to other $k-1$ facilities. See Figure 1.3(b) for an illustration of the definition of the dispersion sum. In Figure 1.3(b), the dispersion sum of the facility at p_1 is $\sum_{i=2}^{k} d(p_1, p_i)$.

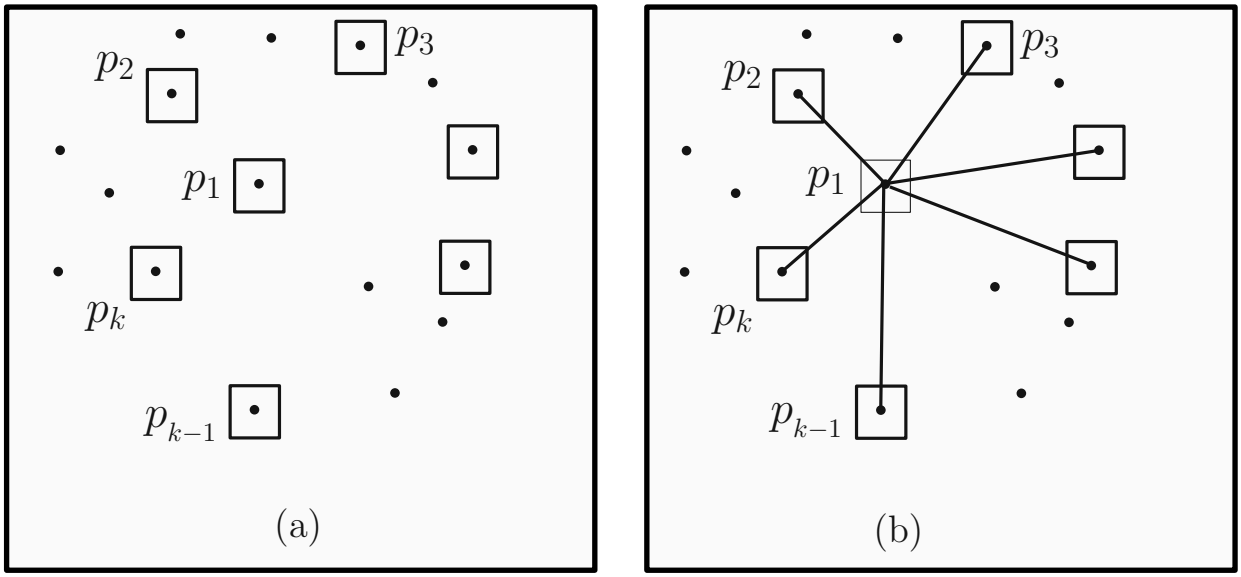


Figure 1.3: *Dispersion sum of facility placed at location* p_1

In the Max-Min-Sum problem, k facilities are to be placed in order to maximize the smallest of the dispersion sum. Note that the 1-dispersion problem is a special case of the Max-Min-Sum problem, where for a facility, the dispersion sum is the distance to the closest placed facility, rather than adding the distances to all other $k-1$ placed facilities. In both Max-Min-Min and Max-Min-Sum problems, for a located facility, we take into account (i) the facility located closest to it and (ii) all other $k-1$ located facilities, respectively. Consequently, one can explore only two extremes of a dispersion spectrum. Therefore, it is reasonable to examine an intermediate metric that considers the effect of some facilities, but

not all. Hence, the concept of *dispersion partial sum*, a more general metric, is introduced. Now, to define the dispersion partial sum, assume that k locations are selected (to place facilities) from n desired locations. Then, for a facility, the dispersion partial sum is the sum of the distance of the pre-specified number of the closest facilities. Hence, the concept of a dispersion partial sum generalizes the Max-Min-Sum dispersion problem. For each facility, if the dispersion partial sum is the sum of the distance of the c ($\leq k$) closest placed facilities, then the Max-Min-Sum problem is referred to as the *c-dispersion* problem. The c-dispersion problem is defined formally as follows.

c-Dispersion Problem: *Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points, the non-negative distance $d(p_i, p_j)$ between each pair of points $p_i, p_j \in P$, and a positive integer k ($c+1 \leq k \leq n$), for each point $p \in P$ and a subset S of P, the c-dispersion cost of the point p with respect to S, $cost_c(p, S)$, is defined as the sum of distances from p to the c closest points in $S \setminus \{p\}$. The c-dispersion cost of the subset S of P is defined as $cost_c(S) = \min_{p \in S}\{cost_c(p, S)\}$. The objective of the c-dispersion problem is to find a subset $S \subseteq P$ of size k such that $cost_c(S)$ is maximized.*

Motivations. Consider a scenario in which a franchiser wants to open six stores in a city (out of n desired locations) so that the stores are far from each other. Now, if the stores are opened at these six locations: $p_1, p_2, p_3, p_4, p_5, p_6$, then the dispersion sum metric for the store located at p_1 gives the false impression that all the stores at p_2, p_3, p_4, p_5 and p_6 are scattered from the store located at p_1 (see Figure 1.4 (a)). This is because the stores located at p_2, p_4 and p_6 are far from the location p_1. On the other hand, the stores located at p_3 and p_5 are relatively close to the store located at p_1 (see Figure 1.4(b)). For the scenario considered in Figure 1.4, to get a better idea of the dispersion, it would be wise to look at

the two closest stores. Thus, to precisely measure the dispersion, it would make more sense to consider a fixed number of the closest open stores using the dispersion partial sum as a metric. See Figure 1.4(b), where the two closest stores of p_1, *i.e.*, p_3 and p_5, are considered.

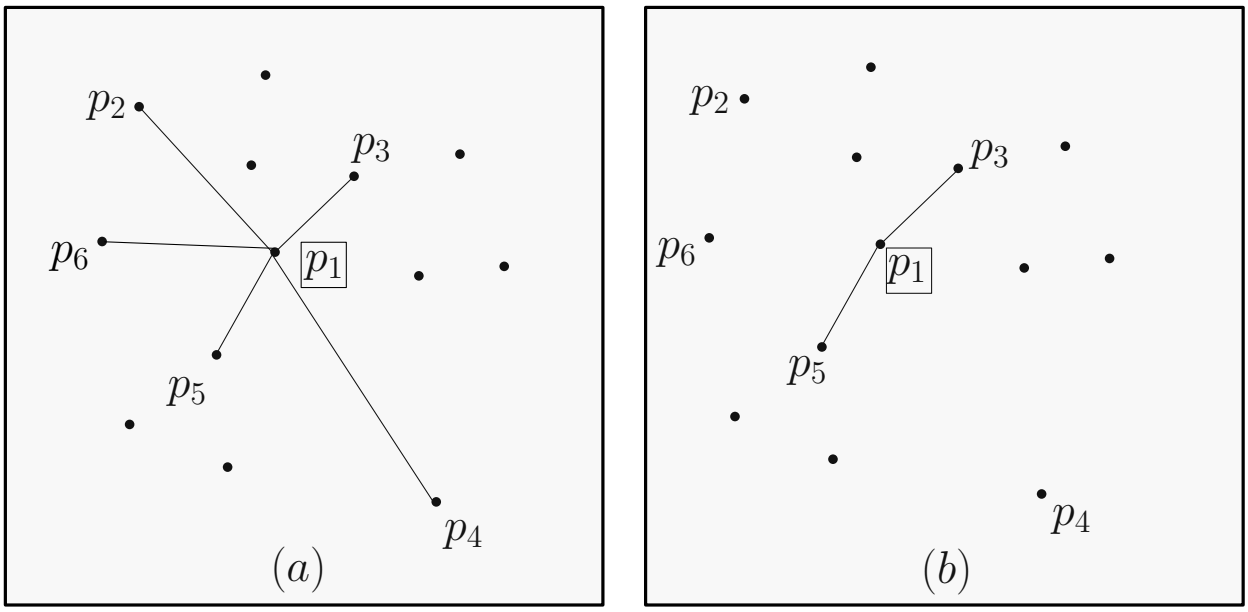


Figure 1.4: *(a) dispersion sum metric, and (b) dispersion partial sum metric*

Based on the scenario discussed in the example in Figure 1.4, where it was shown that for a facility, the two closest facilities are relevant to effectively comprehend the dispersion, we introduce another variant of the Max-Min-Sum dispersion problem, called the 2-dispersion problem. The 2-dispersion problem is defined as follows.

2-Dispersion Problem: *Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points, the non-negative distances between each pair of points $p_i, p_j \in P$, and a positive integer k $(3 \leq k \leq n)$, for each point $p \in P$ and a subset S of P, the 2-dispersion cost of the point $p \in P$ with respect to S, $cost_2(p, S)$, is defined as the sum of Euclidean distances from p to the closest point in $S \setminus \{p\}$ and the second closest point in $S \setminus \{p\}$. The 2-dispersion cost of the subset S of P is defined as $cost_2(S) = \min_{p \in S}\{cost_2(p, S)\}$. The objective of the 2-dispersion problem is to find a subset $S \subseteq P$ of cardinality k such that $cost_2(S)$ is maximized.*

To this point, in all the dispersion problems that have been considered, a given set of locations is regarded as an arbitrary set of points on the plane. However, a set of locations can be restricted to certain geometric objects, such as lines, circles, convex polygons, *etc.*. Now consider a variant of the 1-dispersion problem, where a set of locations is the vertices of a convex polygon. This variant of the 1-dispersion problem is referred to as the *convex 1-dispersion* problem, and it is defined as follows.

Convex 1-Dispersion Problem: *Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n vertices of a convex polygon, the Euclidean distance $d(p, q)$ between each pair of vertices $p, q \in P$, the objective of the convex 1-dispersion problem is to find a subset $S \subseteq P$ of vertices of size k such that the cost of a subset S, $cost(S) = \min\{d(p, q) \mid p, q \in S\}$, is maximized.*

Motivations. The inspiration for studying the problem came when we tried to open chain stores for a large amusement park. Consider the following scenario: we have a park whose geographical boundary forms a convex polygon C (see Figure 1.5), and the preferred locations for opening food chain stores are located only on the vertices of C. Let the number of

preferred locations be n. The objective is to open k food chain stores out of all preferred n locations so that the stores are mutually far from each other. Basically, we need to eliminate or avoid self-competition among stores.

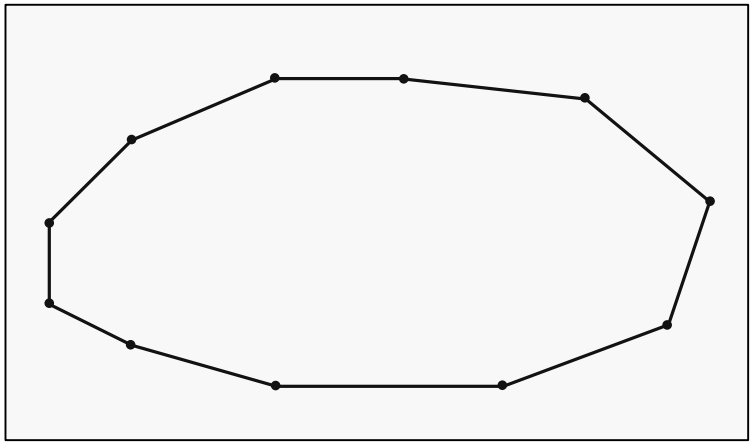


Figure 1.5: *Convex Polygon*

1.1 Preliminaries

The complexity class P comprises problems for which a deterministic polynomial-time algorithm exists. In computational complexity theory, the notion of polynomial-time leads to several complexity classes. One of the most important classes defined using polynomial-time is NP (nondeterministic polynomial time). NP is the complexity class of decision problems[1] that can be solved on a non-deterministic Turing machine in polynomial time. The two main types of problem widely studied with respect to class NP are:

NP-hard problem: A problem H is NP-hard when every problem L in NP can be reduced in polynomial-time to an instance of H; that is, that is, assuming a solution for the reduced instance of H one can produce the solution of the problem L in polynomial time.

[1]A decision problem is a computational problem that can be posed as a yes–no question of the input values.

NP-complete problem: A problem is NP-complete when it is both NP-hard and in NP.

The overwhelming majority of interesting optimization problems are NP-hard. Therefore, we consider designing approximation algorithms for these problems.

Approximation algorithm: An approximation algorithm A for a given problem P finds a solution close to the optimal solution, with provable guarantees on how far the returned solution is to the optimal one. Note that the algorithm A runs in polynomial-time for all instances of the problem. An algorithm A is a ρ factor approximation for the problem P if, for all instances of the problem P, an algorithm A produces a feasible solution whose value is within ρ-factor of the optimal solution value.

Polynomial-Time Approximation Scheme (PTAS): An algorithm A is said to be *polynomial-time approximation scheme* ($PTAS$) for a minimization (resp. maximization) problem P if, for any instances of P and given any rational value $\epsilon > 0$, A is a $(1+\epsilon)$ (resp. $1-\epsilon$)-factor approximation algorithm for the problem P. Note that the running time of A must be polynomial in the input size of the problem, but not necessarily in $1/\epsilon$. An algorithm A is said to be *fully polynomial-time approximation scheme* ($FPTAS$) if the running time of A is required to be polynomial both in the input size of the problem and in $1/\epsilon$.

We define common notion in Parameterized Complexity here.

Parameterized Problem: A parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed and finite alphabet. Any $(I,k) \in \Sigma^* \times \mathbb{N}$ is called instance of L. (I,k) is called a Yes instance of L if $(I,k) \in L$, otherwise called a No instance. For an instance (I,k), k is called the parameter.

Next, we define fixed-parameter tractability.

Fixed-Parameter Tractability: Let $L \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. We say that L is a fixed-parameter tractable (FPT) if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$, a constant c and an algorithm A that takes as input an instance (I, k) of L, runs in time $f(k).|I|^c$ and correctly decides if $(I, k) \in L$.

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

- (x, k) is an Yes instance of A if and only if (x', k') is an Yes instance of B
- $k' \leq g(k)$ for some (non-decreasing) computable function g, and
- the running time is $f(k)|x|^{O(1)}$ for some (non-decreasing) computable function f.

If there is a parameterized reduction from A to B, and B is FPT, then A is also FPT (Theorem 13.2 [28]).

W-hardness: Downey and Fellows [32] introduced the notion of W-hierarchy to classify the hardness of parameterized problems. It is sufficient for our thesis goals to state that we have the following hierarchy: $FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots$. Similar to the NP-hardness, if there is a parameterized reduction from a $W[t]$-hard problem A to a parameterized problem B, then B is $W[t]$-hard. Under the assumption that $FPT \neq W[1]$, we can conclude that a $W[1]$-hard problem is not FPT.

1.2 Scope of the Thesis

In this thesis, we consider the facility location problem, more specifically the covering and dispersion problems. We study both problems because of their wide applications in the

real world. See [1, 16, 19, 53, 56, 57] and [18, 64, 70, 75] for some applications of the covering problem and the dispersion problem, respectively. Most of the problems that we consider in this thesis are NP-hard, and the status of the hardness of other problems is unknown. We study a variant of the covering problem, namely the (α, P, Q)-covering problem. We study variants of the dispersion problem in Euclidean space, namely the 2-dispersion problem in $\mathbb{R}^1$ and $\mathbb{R}^2$, and the 1-dispersion problem in $\mathbb{R}^2$. Furthermore, we study the dispersion problem in a metric space. In a metric space, we introduce a variant of the dispersion problem, namely, the c-dispersion problem. We further study the dispersion problem for the restricted input, where given set of points are in convex position. We refer to it as the convex 1-dispersion problem. For the (α, P, Q)-covering problem, we establish a necessary and sufficient condition that ensures the feasibility of the given instance, prove that the (α, P, Q)-covering problem is NP-complete for $\alpha \geq 3$, and also propose an algorithm that admits a PTAS. For the 2-dispersion problem in $\mathbb{R}^2$, we propose a simple polynomial-time algorithm that produces the $(2\sqrt{3} + \epsilon)$-factor approximation result, for any $\epsilon > 0$. We improve the approximation factor to $2\sqrt{3}$ in $\mathbb{R}^2$. We also present a polynomial-time algorithm for the 2-dispersion problem in $\mathbb{R}^1$ that returns an optimal solution. We further propose a 2-factor approximation algorithm for the 1-dispersion problem in $\mathbb{R}^2$. Next, for the c-dispersion problem in a metric space, we present a polynomial-time algorithm which yields a $2c$-factor approximation result. We also show that the c-dispersion problem in a metric space parameterized by the solution size k is $W[1]$-hard. Finally, we study the convex 1-dispersion problem. We propose an $O(n^3)$-time algorithm that returns an optimal result where the objective is to select $k(= 4)$ vertices. We also propose a $\sqrt{3}$ (≈ 1.733)-factor approximation algorithm for the convex 1-dispersion problem.

1.3 Organization of the Thesis

The rest of this thesis is organized as follows:

Chapter 2: Literature Review. In this chapter, we discuss existing work on covering and dispersion problems.

Chapter 3: Capacitated Discrete Unit Disk Cover Problem. In this chapter, we start with a description of the (α, P, Q)-covering problem. We establish a necessary and sufficient condition through which one can ensure whether the given instance is feasible or not. Furthermore, we prove that the problem is NP-complete for $\alpha \geq 3$ and propose a local search algorithm that admits a PTAS for the (α, P, Q)-covering problem.

Chapter 4: Euclidean Dispersion Problem. In this chapter, we present a $(2\sqrt{3}+\epsilon)$-factor approximation result for the 2-dispersion problem in $\mathbb{R}^2$. We also develop a common framework for the dispersion problem in Euclidean space, which returns a $2\sqrt{3}$-factor approximation result and an optimal result for the 2-dispersion problem in $\mathbb{R}^2$ and $\mathbb{R}^1$, respectively. Using the common framework, we propose a 2-factor approximation result for the 1-dispersion problem in $\mathbb{R}^2$.

Chapter 5: Dispersion Problem in a Metric Space. In this chapter, we study the c-dispersion problem in a metric space and propose a greedy algorithm that produces a $2c$-factor approximation result. We also prove that the c-dispersion problem in a metric space parameterized by the solution size is W[1]-hard.

Chapter 6: Convex 1-Dispersion Problem. In this chapter, we study the convex 1-dispersion problem. We study a variant in which the objective is to select the $k(=4)$ vertices of a convex polygon. We propose an iterative algorithm that produces an optimal solution in $O(n^3)$ time. In addition, we also propose a $\sqrt{3}$-factor approximation result for the convex 1-dispersion problem for any value of k.

Chapter 7: Conclusion and Future Work. In this chapter, we discuss concluding remarks of the thesis and some open problems which can be considered for future research.

2

Literature Review

In this chapter, we discuss the state-of-the-art results for the covering and dispersion problems. We discuss the existing results for both problems in both general and geometric settings. The chapter is divided into two sections. In Section 2.1, we discuss the state-of-the-art results for the covering problem. In Section 2.2, we study the existing results for the dispersion problem.

2.1 Covering Problem

The set cover (SC) problem is one of the most popular covering problems. In 1974, Johnson [58] proposed a greedy algorithm for the SC problem that produces an $O(\log n)$-factor approximation result. Moreover, unless P=NP, the approximation factor for the SC problem cannot be improved [9, 31, 42]. The vertex cover (VC) problem is another interesting variant of the covering problem which has been extensively studied. Unless P=NP, Safra and Dinur [30] showed that the VC problem cannot be approximated within a factor of 1.36.

In 1982, Wolsey [81] proposed a greedy algorithm for the hard capacitated version of the SC problem and achieved a $O(\log n)$-factor approximation result. In 2003, Guha *et al.* [52] introduced the soft capacitated VC problem. An application in the field of glycology has motivated Guha *et al.* to study the soft capacitated VC problem. The problem was based on research done by Glycodata, a biotechnology company. They formulated the problem in ILP and used a simple LP rounding scheme on the relaxed LP and proposed a 4-factor approximation result. They improved the factor to 2 using a primal dual approach. Gandhi *et al.* [47] proposed a 2-factor approximation algorithm for the soft capacitated VC problem using dependent rounding. Chuzhoy and Naor [25] were the first to address the hard capacitated VC problem. They discussed both weighted and unweighted versions of the hard capacitated VC problem. They showed that for the weighted hard capacitated VC problem, it is NP-hard to approximate within a factor of $(1 - \epsilon) \log n$. They also presented an algorithm with an approximation factor 3 for the unweighted version. In 2006, Gandhi *et al.* [46] studied the hard capacitated VC problem and proposed a 2-factor approximation result. Although the weighted version of the hard capacitated VC problem is NP-hard to approximate within a factor of $(1 - \epsilon) \log n$, Gandhi *et al.* [47] proposed a 2-factor approximation algorithm by allowing each vertex to be used at most twice the given bound. In

2012, Saha and Khuller [72] were the first to define the hard capacitated VC problem for hypergraphs and multigraphs. They obtained a 38-factor approximation result for multigraphs and a $\max\{6f, 65\}$-approximation result for hypergraphs, where each hyperedge is of size at most f. They proposed a method based on LP rounding for both problems. In 2014, Cheung et al. [24] gave a 2.155-factor approximation algorithm for multigraphs and a $2f$-factor approximation algorithm for hypergraphs for the hard capacitated VC problem. Inspired by the above result, in 2017, Wong [82] proposed a 2-factor approximation algorithm and a f-factor approximation algorithm for multigraphs and hypergraphs, respectively. The algorithms for both problems are based on iterative rounding to natural LP relaxation. It is known for the VC problem for hypergraphs, the best-known approximation ratio is also f, as it cannot be improved even further according to the unique game conjecture.

Now, we discuss the state-of-the-art results for the SC problem in geometric settings. Recent research has emphasized the geometric set cover (GSC) problem, which refers to the situation in which there are no limitations on the number of elements a geometric object can cover. One of the most studied GSC problems is the *Discrete Unit Disk Cover* (DUDC) problem, where a set of n points and a set of m unit disks are given as input. It is a well-studied problem and has wide application in wireless networks [29]. Note that the (α, P, Q)-covering problem is the same as the DUDC problem, for $\alpha = |P|$. The DUDC problem is NP-complete [48]. The first constant factor approximation algorithm for the DUDC problem was proposed by Brönnimann and Goodrich [14]. They made an interesting relation between the DUDC problem and the ϵ-net [1]. Brönnimann and Goodrich used the theorem proposed by Haussler and Welz, which states that for range spaces with VC dimension d, there exists an ϵ-net of size $O(\frac{d}{\epsilon} log \frac{d}{\epsilon})$ [55]. The constant factor approximation algorithm proposed

[1]For a given range space $S = (X, \mathscr{R})$, an ϵ-net is a subset $P \subseteq X$ such that $P \cap R \neq \phi$ for all $R \in \mathscr{R}$ with $|R| \geq \epsilon n$.

by Brönnimann and Goodrich depends on the large constant value in the size of ϵ-net. Subsequently, many constant-factor approximation results have been published using various techniques. In 2004, Călinescu *et al.* [15] proposed a 108-factor approximation algorithm that runs in polynomial-time. They obtained a 108-factor approximation algorithm for the DUDC problem by decomposing the instance. They derived an algorithm to find the unit disks required to cover points in an equilateral triangle with unit length sides. Furthermore, they gave a 6-factor approximation algorithm to find the unit disks required to cover each equilateral triangle. Also, they showed that a unit disk covers points with no more than 18 triangles, giving a total approximation factor of 108. In 2006, Ambühl *et al.* [5] improved the approximation factor to 72 in $O(m^2)$ time. They obtained a 2-factor approximation algorithm to cover points in squares of size $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$. In total, it is required to compute it for 36 such squares, which leads to 72-factor approximation algorithm. In 2007, Carmi *et al.* [17] further improved the approximation factor from 72 to 38 with an increase in the running time from $O(m^2)$ to $O(m^6)$. This algorithm decomposes planes into squares of size $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$. They used a different variety of procedures to cover points in the square using disks. In 2010, Claude *et al.* [27] improved the approximation factor result and proposed a 22-factor approximation algorithm with the running time of $O(m^2n^4)$. This was possible because of the improved algorithm for Line-Separable DUDC [1]. Using the idea of Ambühl *et al.* [5] and Claude *et al.* [27], Das *et al.* [29] obtained an 18-factor approximation algorithm. The running time of the algorithm is $O(n \log n + m \log m + mn)$, which is a significant improvement compared with other previously proposed algorithms. A year later, in 2012, Fraser *et al.* [44] proposed a 15-factor approximation algorithm for the DUDC problem. They used the decomposition of Das *et al.* [29] to obtain the 15-factor approximation result.

[1] In the Line-separable DUDC problem, the plane is divided by a line ℓ into two halves ℓ^+ and ℓ^-, and all the points in a given set P are in the plane ℓ^- and the center of unit disks are in $\ell^+ \cup \ell^-$ such that each point in P are covered by at least one disk centered in ℓ^+ [29].

In 2013, Manjanna *et al.* [7] proposed a $(9+\epsilon)$-factor approximation algorithm, which runs in $O(\max(m^6 n, m^{2(1+6/\epsilon)+1}))$ time. In 2009, Mustafa and Ray [68] obtained a PTAS using a local search algorithm [1]. The time complexity of the local search algorithm is $O(mn^{O(\epsilon^{-2})})$. Compared with other previously proposed constant factor algorithms, the time complexity of this local search algorithm is extremely high. The algorithm can be easily implemented due to its simplicity. This is one of the advantages of using a local search algorithm. This algorithm is also applicable to disks of any arbitrary radius. It also gives a PTAS for arbitrary radius disks. The GSC problem in which geometric objects other than disks have also been studied extensively; see [6, 20, 26, 51, 67, 78].

2.2 Dispersion Problem

In 1977, Shier [73] introduced the 1-dispersion problem on a tree network and established a relation between the 1-dispersion problem and the k-center problem. Shier showed that the 1-dispersion problem and the $(k-1)$-center problem are dual problems and established an equivalence between both problems. In 1981, Chandrasekaran and Daughety [22] studied the 1-dispersion problem on a tree network and proposed a polynomial-time algorithm. In 1982, Chandrasekaran and Tamir [23] also studied the 1-dispersion problem and the k-center problem on a tree network. They showed that if the set of locations is a finite subset of the continuum set of points on the edges, then there exists an equivalence between the 1-dispersion problem and the $(k-1)$-center problem on a tree network. So, using a k-center algorithm on a tree (proposed in [45]), a linear-time algorithm can be devised for the 1-dispersion problem on a tree. In 1990, Erkut [36] proved that the 1-dispersion problem is NP-hard by showing a reduction from the clique problem. In 1991, White [80] studied the

[1] The local search algorithm is an iterative algorithm that starts with a feasible solution and improves the solution after each iteration until a locally optimal solution is reached.

1-dispersion problem and proposed a 3-factor approximation algorithm. In 1991, Tamir [76] studied the 1-dispersion problem on a general graph, where a continuum set of points on the edges are considered as locations, and showed that if a continuum set of locations on a graph is given, then the 1-dispersion problem cannot be approximated within a factor of $\frac{3}{2}$ unless $P = NP$. Tamir [76] also proposed a heuristic that yields a 2-factor approximation result for the 1-dispersion problem on a graph. Later in 1994, Ravi *et al.* [70] studied the 1-dispersion problem on complete graphs, where each edge is associated with a non-negative weight (distance that maintains the triangle inequality). They independently analyzed the same heuristic proposed in [76] (for the 1-dispersion problem on complete graphs) and showed that the same heuristic produces a 2-factor approximation result for complete graphs. Furthermore, they also demonstrated that, unless $P = NP$, the 1-dispersion problem on complete graphs does not have a better than 2-factor approximation result. In 1991, Megiddo and Tamir [65] designed an $O(k^2 \log^2 n)$ time algorithm for the k-center problem on a line when points are ordered. Note that the same algorithm can be adapted to solve the 1-dispersion problem on a line (the points are not necessarily ordered) in polynomial time. In 2007, Bhattacharya and Shi [10] proposed a linear-time algorithm for the k-center problem on a line, where the points are not necessarily ordered. This algorithm can be adapted to solve the 1-dispersion problem on a line (the points are not necessarily ordered) in polynomial time. Wang and Kuo [79] introduced the 1-dispersion problem in geometric settings. They considered the problem in a d-dimensional space with a Euclidean distance function between two points and proposed a dynamic programming algorithm that solves the problem in $O(kn)$ time for $d = 1$. Furthermore, they proved that for $d = 2$, the problem is NP-hard. Recently, in 2018, Akagi *et al.* [2] established a relation between the 1-dispersion problem and the independent set (IS) problem, and proposed an exact algorithm for the 1-dispersion problem. They proposed an $O(n^{wk/3} \log n)$ time algorithm, where $w < 2.373$. In [2], they

also studied two special cases of the 1-dispersion problem where the set of points (i) lies on a line, and (ii) lies on a circle. They proposed a polynomial-time exact algorithm for both special cases.

The max-sum dispersion problem is another popular variant of the dispersion problem. Here, the objective is to maximize the sum of the distances between the k facilities. Tamir [76] showed that the problem on a line has a trivial solution in $O(n)$ time. He also proved that the problem can be solved in $O(kn)$ time if the points are on a tree. Later, Ravi et al. [70] studied the problem on a line independently and gave a $O(\max(kn, n \log n))$ time algorithm. They also proposed a 4-factor approximation algorithm if the distance function satisfies the triangle inequality and also presented a $(1.571 + \epsilon)$-factor approximation algorithm when the vertices are points on the 2-dimensional Euclidean space and the Euclidean distance between two points is the weight of the corresponding edges, where $\epsilon > 0$. In [13] and [54], the approximation factor of 4 improved to 2. One can see [8, 21, 35, 37, 38, 49, 66] for other variants of the dispersion problem.

Compared to the 1-dispersion problem, a handful of research has been done on the c-dispersion problem. In 2013, Lei and Church [63] introduced the c-dispersion problem, and in 2015 an efficient formulation of the c-dispersion problems was proposed [64]. In 2018, Amano and Nakano [3] proposed a greedy algorithm for the 2-dispersion problem in a metric space. They have shown that the proposed greedy algorithm produces an 8-factor approximation result for the 2-dispersion problem in a metric space. In [3], they also proposed a $2c^2$-factor approximation algorithm for the c-dispersion problem in a metric space. In 2020, Amano and Nakano [4] analyzed the same greedy algorithm proposed in [3], and showed that the greedy algorithm produces a $4\sqrt{3}$ (≈ 6.92)-factor approximation result for the 2-dispersion problem when the distance function between two points is the Euclidean distance.

The design of an efficient algorithm for the convex 1-dispersion problem is open [2]. Recently in 2022, Kobayashi *et al.* [61] studied the convex 1-dispersion problem, where the objective is to select $k = 3$ vertices and proposed an $O(n^2)$ time algorithm to calculate the optimal value. Now, we consider a variant of the convex 1-dispersion problem, where the set of locations is a set of points inside a polygon and vertices of a polygon, and the objective is to select $k = 3$ points. In this problem, the goal is to find a triangle inside the convex polygon whose length of the smallest side is maximum. In 2020, Sadhu *et al.* [71] studied this problem and proposed an $O(n^2)$ time algorithm to select three points.

3

Capacitated Discrete Unit Disk Cover Problem

In this chapter, we study the geometric capacitated set cover (GCSC) problem, where a set of unit disks is considered as a set of geometric objects and the capacity of each unit disk is uniform. We refer to such a GCSC problem as the (α, P, Q)-covering problem, and it is defined as follows:

(α, P, Q)-Covering Problem: *For a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points and a set $Q = \{q_1, q_2, \ldots, q_m\}$ of m points and a positive integer α, a subset $Q' \subseteq Q$ is said to be an α-cover of P if (i) the point set P can be partitioned into ℓ subsets $P_1, P_2, \ldots, P_\ell$ such that $|P_i| \leq \alpha$ for each $i = 1, 2, \ldots, \ell$, where $\ell = |Q'|$ and (ii) each $p \in P_i$ is covered by a unit disk centered at $q'_i \in Q'$. Given a positive integer α, a set P of n points and a set Q of m points, the objective of the (α, P, Q)-covering problem is to find a minimum cardinality α-cover $Q' \subseteq Q$ of P.*

3.0.1 Overview of the Chapter

Goal of the Chapter. To (i) establish a necessary and sufficient condition through which one can ensure the feasibility of the given instance of the (α, P, Q)-covering problem, (ii) prove that the (α, P, Q)-covering problem is NP-complete for $\alpha \geq 3$, and (iii) design a polynomial-time approximation scheme (PTAS) for the (α, P, Q)-covering problem.

Organization of the Chapter. The remainder of the chapter is organized as follows. In Section 3.1, a necessary and sufficient condition is established that ensures the feasibility of the given instance of the (α, P, Q)-covering problem. In Section 3.2, we prove that the (α, P, Q)-covering problem is NP-complete for $\alpha \geq 3$. Furthermore, in Section 3.3, we propose a PTAS for the same problem and finally conclude the chapter in Section 3.4.

3.1 A Necessary and Sufficient Condition

In this section, we establish a necessary and sufficient condition to check the feasibility of an instance of the (α, P, Q)-covering problem, *i.e.*, to check if there exists an α-cover for a set P with respect to a set Q. To state the condition, we construct a bipartite graph

$G = (V_1 \cup V_2, E)$ for a given instance of the (α, P, Q)-covering problem, where V_1 is the set of vertices corresponding to each point in Q and V_2 is the set of vertices corresponding to each point in P and $E = \{e = v_iv_j \mid v_i \in V_1$ and $v_j \in V_2$, and a unit disk centered at q_i covers $p_j\}$. Note that $P = \{p_1, p_2, \ldots, p_n\}$ and $Q = \{q_1, q_2, \ldots, q_m\}$. We define the open neighborhood of a vertex $u \in V_2$, denoted by $N_G(u)$, as the set of open neighbors of u in G. Similarly, we define the neighborhood function for a subset $B \subseteq V_2$, denoted by $N_G(B)$, as $N_G(B) = \bigcup_{u \in B} N_G(u)$.

Theorem 3.1.1. An α-cover for a set P with respect to a set Q exists if and only if for any subset $V_2' \subseteq V_2$, $|N_G(V_2')|\alpha \geq |V_2'|$.

Proof. (**Necessity.**) Suppose that there is an α-cover of P with respect to Q. Let $Q'(\subseteq Q)$ be an α-cover of P with respect to Q. Now, assume for a subset $V_2' \subseteq V_2$, $|N_G(V_2')|\alpha < |V_2'|$. Since $|N_G(V_2')|$ and $|V_2'|$ are positive integers, therefore, there exist at least $|V_2'| - |N_G(V_2'')|$ points of P corresponding to the vertices of V_2' that cannot be covered by any point of Q'. Therefore, it is a contradiction to the assumption that Q' is an α-cover of P with respect to Q.

(**Sufficiency.**) Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points, a set $Q = \{q_1, q_2, \ldots, q_m\}$ of m points and a positive integer α, we construct another bipartite graph $G' = (V_1' \cup V_2, E)$, where $V_1' = \{v_{ij} \mid 1 \leq i \leq m$ and $1 \leq j \leq \alpha\}$ is a set of vertices corresponding to the set of points in Q such that for each point $q_i \in Q$, we consider α vertices in V_1', *i.e.*, v_{ij} $(j = 1, 2, \ldots, \alpha)$ and V_2 is the set of vertices corresponding to the points in P and $E = \{e = v_{ij}v_\ell \mid v_{ij} \in V_1'$ and $v_\ell \in V_2$, and a unit disk centered at q_i covers $p_\ell\}$ (see Figure 3.1). Observe that $|N_{G'}(B)| = |N_G(B)|\alpha$ for any $B \subseteq V_2$. Suppose that for each subset $V_2' \subseteq V_2$, $|N_G(V_2')|\alpha \geq |V_2'|$. This leads to $|N_{G'}(V_2')| \geq |V_2'|$ for each subset $V_2' \subseteq V_2$. Therefore, G' has a matching M of size $|V_2|$ (Hall's Theorem). For each vertex $v_{ij} \in V_1'$ of

the matching M, if we consider all the points corresponding to vertex $v_i \in V_1$, we get an α-cover of a set P with respect to a set Q. □

The algorithm to find a feasible solution is based on the maximum matching algorithm in a bipartite graph. Here, we construct the bipartite graph from any arbitrary instance of the (α, P, Q)-covering problem, which leads us to find a feasible solution in polynomial time. We construct the bipartite graph in a similar approach to G' (the sufficiency part of Theorem 3.1.1). Therefore, the total number of vertices in the bipartite graph G' is $|V| = |V_1| + |V_2| = \alpha m + n$ and the maximum possible number of edges $|E| = \alpha mn$. Note that the construction of the bipartite graph takes $O(\alpha mn)$ time. If the size of the maximum matching in the bipartite graph G' is exactly n (*i.e.*, if the condition of Theorem 3.1.1 is true), then the input instance has a feasible solution.

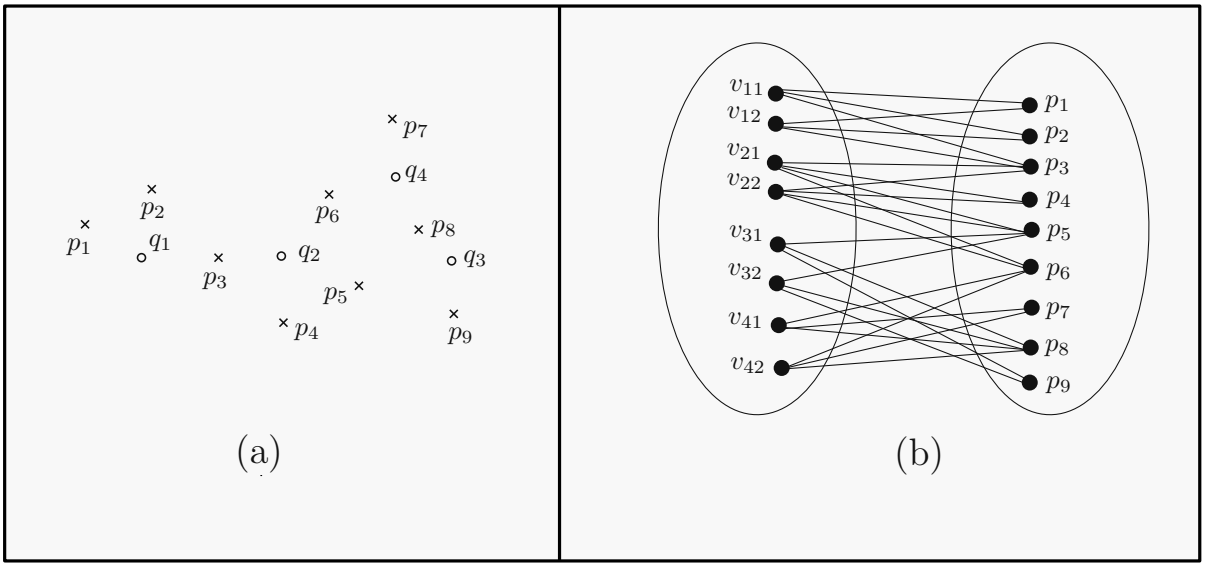


Figure 3.1: *(a) An instance of the (α, P, Q)-covering problem, and (b) Construction of bipartite graph for $\alpha = 2$, here vertex v_{ij} represent j^{th} copy of disk i.*

3.2 Hardness of the (α, P, Q)-Covering Problem

In this section, we show that the (α, P, Q)-covering problem is NP-complete for $\alpha = 3$. Using the NP-complete proof for the (α, P, Q)-covering problem for $\alpha = 3$, we can conclude that the (α, P, Q)-covering for $\alpha \geq 4$ is also NP-complete. Next, we show that the $(3, P, Q)$-covering problem is NP-complete by proving that the $(3, P, P)$-covering problem is NP-complete. Note that the $(3, P, P)$-covering problem is a special case of the $(3, P, Q)$-covering problem.

We show a polynomial-time reduction from the decision version of the vertex cover (VC) problem on a planar graph of degree at most 3 to the decision version of the $(3, P, P)$-covering problem. This reduction will ensure that the $(3, P, P)$-covering problem is NP-complete. Note that the vertex cover problem on a planar graph of degree at most 3 is known to be NP-complete [48].

Decision version of the VC problem on a planar graph (VC-PLA)

Instance: $\langle G, k \rangle$, where $G = (V, E)$ is an undirected planar graph with maximum degree 3, and k is a positive integer.

Question: Does there exist a vertex cover $V'(\subseteq V)$ of G such that $|V'| \leq k$?

Decision version of the $(3, P, P)$-covering problem

Instance: $\langle P, k' \rangle$, where P is a set of points in $\mathbb{R}^2$, and k' is a positive integer.

Question: Does there exist a 3-cover $P'(\subseteq P)$ of P with respect to P such that $|P'| \leq k'$?

To prove the NP-completeness of the $(3, P, P)$-covering problem, we first construct an instance $\langle P, k' \rangle$ of the $(3, P, P)$-covering problem from a given arbitrary instance $\langle G = (V, E), k \rangle$ of the planar vertex cover problem such that there exists a vertex cover $V'(\subseteq V)$ of G satisfying $|V'| \leq k$ if and only if there exists a 3-cover $P'(\subseteq P)$ of P with respect to P satisfying $|P'| \leq k'$.

We construct $\langle P, k' \rangle$ from $\langle G = (V, E), k \rangle$ as follows: we first embed the planar graph $G = (V, E)$ into a planar grid using Lemma 3.2.1 and Corollary 3.2.1.1. Subsequently, we construct P in four steps, where we add a set of points at each step. We introduce points in the planar embedding in each step so that exactly two consecutive points are within a unit distance apart (see Figure 3.4(b)). For details, see Steps 1, 2, 3, and 4 of the proof of Lemma 3.2.2.

Lemma 3.2.1 ([77])**.** Given a planar graph $G = (V, E)$ with maximum degree 4, the graph G can be embedded in the plane such that its vertices are in integer coordinates and its edges are line segments of the form $x = i$ or $y = j$, for integers i and j.

We can embed the planar graph as stated in Lemma 3.2.1 in linear time with at most two bends along each edge [12]. See Figure 3.2(b) for a planar embedding of the planar graph.

Corollary 3.2.1.1. Given a planar graph $G = (V, E)$ with maximum degree 3, the graph can be embedded in the Euclidean plane with each of its vertices at $(3i, 3j)$ and its edges as a sequence of line segments on the lines $x = 3i$ or $y = 3j$ for integers i and j.

Lemma 3.2.2. Let $G = (V, E)$ be a planar graph with maximum degree 3. An instance of a $(3, P, P)$-covering problem can be constructed from G in polynomial time.

Proof. We construct an instance of $(3, P, P)$ in the following four steps.

Step 1: Embedding. The instance of G is embedded in the plane using the algorithms proposed in [12]. In the embedding, each edge in G is a sequence of connected line segment(s). The length of the line segments used in this embedding is three units. Let ℓ be the total number of line segments in the embedding. We add a point p_i for each $v_i \in V$ in the embedding and name it *node points* (see Figure 3.2 (b)).

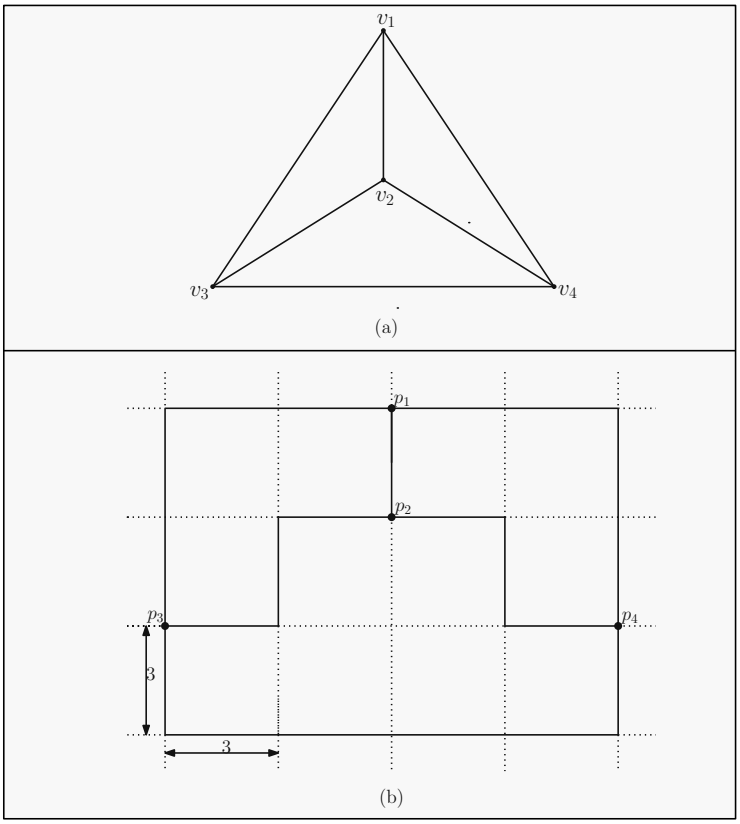


Figure 3.2: *(a) A planar graph G of maximum degree 3, and (b) p_i for each $v_i \in V$.*

Step 2: Adding extra points to the embedding. We classify the line segments in the embedding into two categories, namely, proper and improper. A line segment is *proper* if none of its end points corresponds to node points. Line segments other than the *proper* line segments are named *improper* line segments. For each improper line segment (p_i, p_j) of length 3 units, where both p_i and p_j are node points, we add two points at distances 0.72

and 1.22 units with respect to both p_i and p_j. Thus, we add four points in the process. See the line segment (p_1, p_2) in Figure 3.3(a). For each line segment of length greater than 3 units, we add points as follows: (i) add a point in the joining point (grid point) of each line segment other than the node points and name it as bend points (see empty circular points in Figure 3.3(b)), and (ii) for each improper line segment, we add three points at distances 0.75, 1.5, and 2.25 units from the node point (see the points added with respect to the point p_2 in line segments between p_2 and p_3 in Figure 3.3(b)), and for each proper line segment we add two points at distances 1 and 2 units from its end points, *i.e.*, bend points (see points added between two bend points in line segments between p_2 and p_3 in Figure 3.3(b)). The points added in this step for the line segments between p_2 and p_3 are explained in detail in Figure 3.3(b). We name the points added in this step as *joint points*.

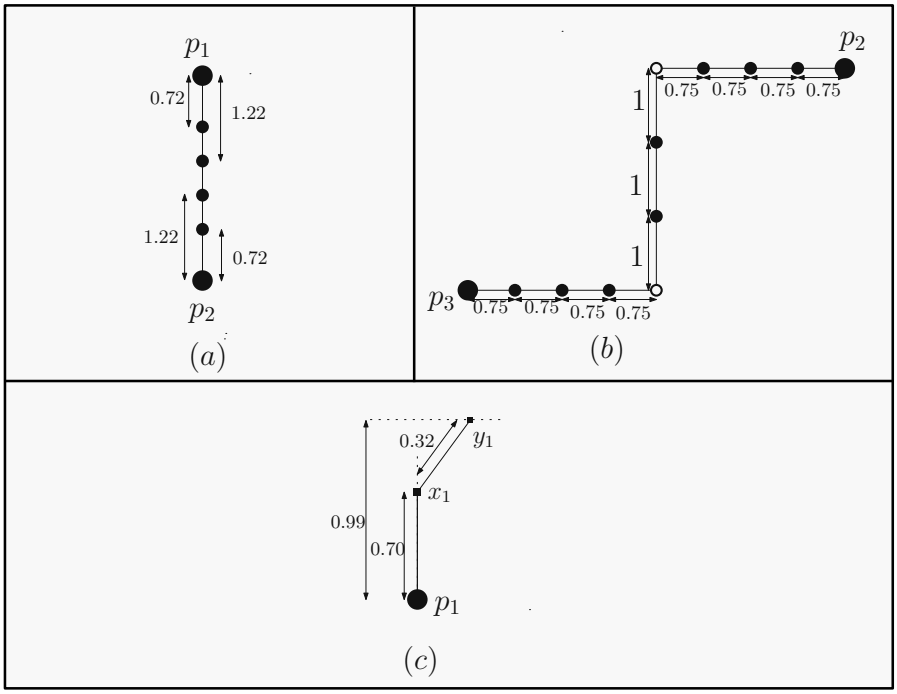


Figure 3.3: *(a) placement of joint points where $\ell' = 1$, (b) Placement of joint points where $\ell' > 1$, and (c) placement of support points with respect to p_1.*

Step 3: Adding extra points. For each node point p_i, add a line segment of 0.70 units of length (on the line $x = 3i$ or $y = 3j$ for some integers i or j), without coincident with other line segments. We can add such line segments on the $x = 3i$ or $y = 3j$ lines without losing planarity, since the maximum degree of G is 3. Now, add a point (say x_i) on the new line segments at a distance of 0.70 units from each node point p_i, and add another point (say, y_i) at a distance of 0.32 units from x_i touching the line at a distance of 0.99 units from p_i. Note that the distance between two points y_i (with respect to p_i) and y_j (with respect to p_i) is greater than one unit. See both the points x_1 and y_1 added with respect to p_1 in Figure 3.3(c). Note that if the Euclidean distance between any two points in the embedding is at most 1 unit, then we can say that these two points are connected. According to the placement of y_i, it is only connected to x_i as the distance of y_i to all other points excluding x_i is greater than one unit. The points added in this step are named *support points*.

Step 4: Construction of P. We denote the set of node points, the set of joint points, and the set of support points by N, J, and S, respectively. Here, $N = \{p_i \mid v_i \in V\}$, $J = \{q_1, q_2, \ldots, q_{3\ell+|E|}\}$ and $S = \{x_i, y_i \mid v_i \in V\}$. We construct $P = N \cup J \cup S$ by removing all line segments from the embedding. See Figure 3.4. Observe that $|N| = |V|(= n)$, $|S| = 2|V|(= 2n)$ and $|J| = 3\ell + |E|$, where ℓ is the total number of line segments in the embedding and $|E|$ is the total number of edges in G. Since G is planar, $|E| = O(n)$. It also follows from Lemma 3.2.1 that $\ell = O(n^2)$. Therefore, P can be constructed in polynomial time.

□

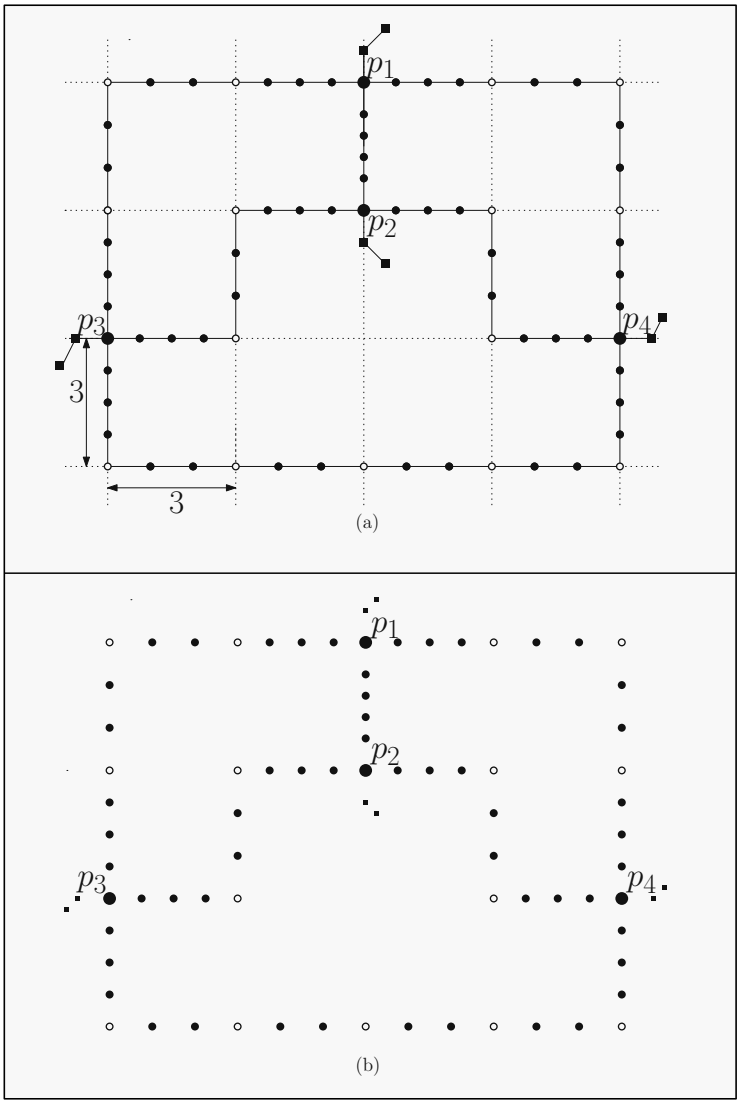


Figure 3.4: *(a) Added points and line segments in the embedding, and (b) instance of the* $(3, P, P)$*-covering problem.*

Theorem 3.2.3. $(3, P, P)$-covering problem is NP-complete.

Proof. For a given set $P' \subseteq P$ and a positive integer k, we can verify whether P' is a 3-cover of P with respect to P itself such that $|P'| \leq k$ in polynomial time. Hence, the $(3, P, P)$-covering problem is in NP. Now, we prove the hardness of the $(3, P, P)$-covering problem by reducing Vc-Pla to it. Let $G = (V, E)$ be an instance of the Vc-Pla. Construct an

instance P of the $(3, P, P)$-covering problem as discussed in Lemma 3.2.2.

Lemma 3.2.4. *G has a vertex cover of size at most k if and only if P has a 3-cover of size at most $k + \ell + n$.*

Proof. **Necessity.** Let $D \subseteq V$ be a vertex cover of G such that $|D| \leq k$. Let $N' = \{p_i \in P \mid v_i \in D\}$, *i.e.*, N' be the set of points in P that correspond to the vertices in D. Let $S' \subseteq S$ be a set of n points chosen from S such that of the two support points associated with each point $p_i \in N$, we select the closest support point x_i in S', *i.e.*, $S' = \{x_i \mid p_i \in N\}$. From each set of points corresponding to a line segment in the embedding, we choose 1 point such that the set of chosen points, say $J'(\subseteq J)$, together with N' and S' will form a 3-cover of desired cardinality in P. Initially, $J' = \emptyset$. As D is a vertex cover, every edge in G has at least one of its end vertices in D. Let (v_i, v_j) be an edge in G and $v_i \in D$ (the tie can be broken arbitrarily if both v_i and v_j are in D). Note that the edge (v_i, v_j) is represented as a sequence of line segments in the embedding. Start traversing the points (of (v_i, v_j)) from p_i, where p_i corresponds to v_i, and add every third point to J' encountered in the traversal without including p_j (see (p_3, p_4) in Figure 3.5 (b), the cross points are part of J' while traversing from p_3). Apply the above process to each edge in G. Observe that the cardinality of J' is ℓ as we choose 1 point from each set of points on a segment in the embedding. Let $P' = N' \cup J' \cup S'$. Now, we argue that P' is a 3-cover of P with respect to P. Each $p_i \in N$ is covered by x_i in S'. Together with p_i, x_i covers itself and y_i. Therefore, each x_i covers at most 3 points and S' ensures coverage for the set S and N. Now, it remains to prove that all the points in J are also covered. For any point $p_i \in N$, there are at most 3 neighbor points of p_i in J (since the maximum degree of G is 3). Choose each point $p_i \in N'$ (*i.e.*, the corresponding point $v_i \in D$ in G), it covers all its neighboring points in J and all other points in J are covered by at least one point $q_j \in J'$. The existence of q_j is guaranteed

by the way we constructed J' and any point $q_j \in J'$ covers at most 3 points, including itself and its neighbors q_{j-1} and q_{j+1}. Therefore, every point in P is covered by at least one point in P', and each point $p_i \in P'$ covers no more than 3 points. Thus, P' is a 3-cover of P and $|P'| = |N'| + |J'| + |S'| \leq k + \ell + n$.

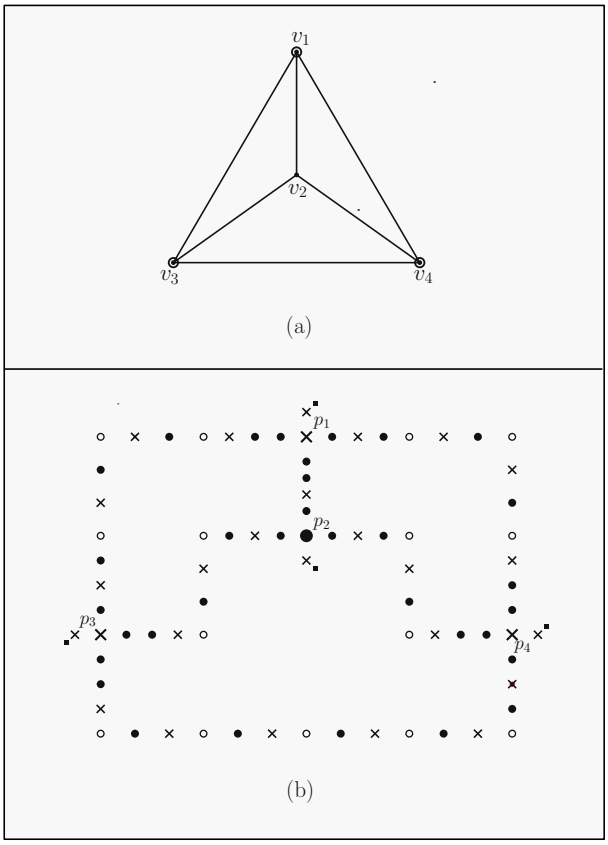


Figure 3.5: *(a) A vertex cover $\{v_1, v_3, v_4\}$ of G, and (b) the construction of J' and S'.*

Sufficiency: Let $P' \subseteq P$ be a 3-cover of size at most $k + \ell + n$. We prove that G has a vertex cover of size at most k with the help of the following claims.

Claim(i): At least one of the support points corresponding to each node point belongs to P'.

Proof of Claim (i): The claim follows from the fact that the support point y_i corresponding to node p_i is covered only by the support points x_i and/or y_i.

Claim(ii): The points corresponding to each segment in the embedding must contribute at least 1 point to P', *i.e.*, $|J \cap P'| \geq \ell$, where ℓ is the total number of segments in the embedding.

Proof of Claim (ii): For each edge (v_i, v_j) in G, there exist some line segments that connect p_i to p_j in the embedding. Let ℓ' be the number of line segments between p_i and p_j, then the total number of points between p_i and p_j is $3\ell' + 3$, including p_i and p_j. Now, if both p_i and p_j are in P', then p_i and p_j can cover themselves along with its neighbor points (say q_i and q_j within these ℓ' segments). So, in the worst case $(3\ell' + 3 - 4) = 3\ell' - 1$ number of points remains to be covered. It needs at least $\left\lceil \frac{3\ell'-1}{3} \right\rceil = \ell'$ number of points. Thus, the claim follows.

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

Proof of Claim (iii): Let (v_i, v_j) be an edge in G such that p_i and p_j are not in P'. By Claim (ii), $|J \cap P'| \geq \ell$. Therefore, the points corresponding to each segment between p_i and p_j representing the edge (v_i, v_j) contribute at least ℓ' points in P'. We argue that if both p_i and p_j are not in P', then the number of points in P' from the points corresponding to each segment representing the edge (v_i, v_j) is at least $\ell' + 1$. Based on our construction of points in the embedding, if there exist ℓ' segments between the points p_i and p_j, then we consider exactly $3\ell' + 1$ points between them. Observe that one point can cover at most 3 points. Thus, to cover $3\ell' + 1$ points, at least $\left\lceil \frac{3\ell'+1}{3} \right\rceil = \ell' + 1$ points are required.

Now, we show that by removing and/or replacing some points in P', a set of k points

from N can be chosen such that the set of corresponding vertices in G is a vertex cover. The points in S' account for the n points in P' (due to Claim (i)). Let $P' = P' \setminus S'$ and $D = \{v_i \in V \mid p_i \in P' \cap N\}$. If any edge (v_i, v_j) in G has none of its end vertices in D, then we do the following: consider the sequence of points corresponding to the segments representing the edge (v_i, v_j) in the embedding. Since both p_i and p_j are not in P', there must exist a segment that has two of its points in P' (due to Claim (iii)). Now consider the points corresponding to the segment that has two points in P'. Delete one of the points in the segment and introduce p_i (or p_j). Update D and repeat the process until each edge has at least one of its end vertices in D. Therefore, D is a vertex cover in G and $|D| \leq k$ (due to claim (ii)). □

By Lemma 3.2.4, we prove that the $(3, P, P)$-covering problem is NP-hard. We have already shown that the $(3, P, P)$-covering problem is in NP. Therefore, the $(3, P, P)$-covering is NP-complete.

□

3.3 A PTAS

We use the local search algorithm to find an α-cover for the (α, P, Q)-covering problem. We prove that the local search algorithm produces a PTAS (see Algorithm 1 for a detailed pseudo-code). Here, we assume that every instance of the (α, P, Q) covering problem has a feasible solution. Furthermore, it can be verified as follows: Actually, the (α, P, Q)-covering problem is a special case of hard capacitated set cover problem, where the bound on the number of available copies of each covering set is at most one. Consider the maximum flow problem for the directed graph $G = (V, E)$ as follows: (i) $V = V_1 \cup V_2 \cup \{s, t\}$, where V_1 and V_2 are the set of vertices corresponding to the points sets Q and P respectively, s is a source

vertex and t is a terminal vertex, and (ii) $E = E_1 \cup E_2 \cup E_3$, where $E_1 = \{(s, v) : v \in V_1\}$, $E_2 = \{(u, v) : u \in V_1, v \in V_2$ and the point corresponding to the vertices u and v are within a unit distance apart$\}$, and $E_3 = \{(u, t) : u \in V_2\}$ such that the capacity of each edge in E_1 is α and the capacity of each edge in $E_2 \cup E_3$ is 1. If the maximum flow in the network mentioned above is exactly n, then the input instance has a feasible solution. Thus, the feasibility of an instance can be checked in $O(n^2m)$ time.

Algorithm 1 Local_Search_Algorithm(α, P, Q)

Input: A set P of n points, a set Q of m points, and a positive integer α.
Output: An α-cover subset $Q'(\subseteq Q)$ of P with respect to Q.

1: $Q' \leftarrow Q$. /*Assume that the given instance has a feasible solution.*/
2: **while** there exist $B \subseteq Q'$ of size at most k and $B' \subseteq Q$ of size at most $k-1$ such that $Q'' = (Q' \setminus B) \cup B'$ is a feasible solution for the (α, P, Q)-covering problem.
3: set $Q' \leftarrow (Q' \setminus B) \cup B'$.
4: **endwhile**
5: Report Q'.

Lemma 3.3.1. The running time of Algorithm 1 is $O(n^2m^{2k})$ for some positive integer k.

Proof. The number of local improvement steps is bounded by the number of points in the set Q. Hence, there is a scope for at most m local improvement steps. In each step, it is required to verify at most $\binom{m}{k}\binom{m}{k-1} \leq m^{2k-1}$ different local improvements (see line 2 in Algorithm 1). The time to check whether a certain local improvement is possible takes $O(n^2m)$ time. Therefore, the overall time complexity of the algorithm is $O(n^2m^{2k})$. □

A subset $Q' \subseteq Q$ is called k-locally optimal if it is not possible to perform the local improvement step. Now, we prove that Algorithm 1 produces a $(1+\epsilon)$-factor approximation result. First, we introduce the notion of *locality condition* before delving into the proof. It is stated as follows.

Locality condition: Let $Q_{opt} \subseteq Q$ be an optimal solution of the (α, P, Q)-covering problem and $Q' \subseteq Q$ be an α-covering set returned by the local search algorithm (Algorithm 1). It is possible to construct a bipartite planar graph $G = (Q' \cup Q_{opt}, E)$ such that for each $p \in P$, there exist two vertices $u \in Q_{opt}$ and $v \in Q'$ share an edge $(u, v) \in E$. Note that $u \in Q_{opt}$ and $v \in Q'$ share an edge if their Euclidean distance is less than or equal to 1.

The locality condition for the range space consisting of points and disks is established in [69]. The locality condition for the (α, P, Q)-covering problem can be established with the help of arguments similar to those in [69]. We define the neighborhood of the vertices in the graph G, *i.e.*, $N_G(u)$ is the set of neighbors of u in the bipartite planar graph and the neighborhood function for the subset Y of the vertices of the graph G, $N_G(Y) = \bigcup_{u \in Y} N_G(u)$.

Lemma 3.3.2. Let $Q_{opt} \subseteq Q$ be an optimal solution and $Q' \subseteq Q$ be returned by Algorithm 1. Assume $Q_{opt} \cap Q' = \phi$ and if there exists a planar bipartite graph $G = (Q' \cup Q_{opt}, E)$, then for every subset $Q'' \subseteq Q'$ of size almost k, $|N_G(Q'')| \geq |Q''|$.

Proof. Let $G = (Q' \cup Q_{opt}, E)$ be a bipartite graph. Since both Q' and Q_{opt} are α-cover sets for the point set P with respect to Q, for each point $p \in P$ there exist a point $q' \in Q'$ and a point $q_{opt} \in Q_{opt}$ such that the Euclidean distance between (i) p and q', and (ii) p and q_{opt} is less than or equal to 1.

Claim 3.3.1. *For any $Q'' \subseteq Q'$, $(Q' \setminus Q'') \cup N_G(Q'')$ is an α-cover of P with respect to Q.*

Proof of Claim. If there is a point $p_i \in P$ that is covered by the unit disk centered at a point in Q'', then one of the neighbors in $N_G(Q'')$ also covers the point p_i due to the locality condition. Therefore, $N_G(Q'')$ covers all the points covered by Q''. So, $(Q' \setminus Q'') \cup N_G(Q'')$ is an α-cover of P with respect to Q.

The above claim implies that if $Q'' \subseteq Q'$ is a set of at most k unit disks, then $|N_G(Q'')| \geq |Q''|$, otherwise, there is scope for a local improvement step.

□

Without loss of generality, assume that $Q_{opt} \cap Q' = \phi$. If not, let $I = Q_{opt} \cap Q'$, $Q^*_{opt} = Q_{opt} \setminus I$, $Q''' = Q' \setminus I$ and let P' be the set of points that are not covered by the disks centered at I. Q^*_{opt} and Q''' are disjoint. Also, Q^*_{opt} is an α-cover of minimum size for the point set P'. Now, we establish a relation between $|Q'''|$and $|Q^*_{opt}|$, which will help us prove that the proposed algorithm admits a PTAS. Now, we state a theorem that helps us establish a relation between $|Q'''|$and $|Q^*_{opt}|$, and it is as follows.

Theorem 3.3.3. [41] For any planar graph $G = (V, E)$ of n vertices, there is a set $X \subseteq V$ of size at most $\frac{c_1 n}{\sqrt{r}}$, such that $V \setminus X$ can be partitioned into n/r sets $V_1, V_2, \ldots, V_{n/r}$ satisfying

i. $|V_i| \leq c_2 r$

ii. $N(V_i) \cap V_j = \phi$ for $i \neq j$, and

iii. $|N(V_i) \cap X| \leq c_3\sqrt{r}$

where $c_1, c_2, c_3 > 0$ and $N(.)$ define the neighborhood function.

Lemma 3.3.4. $|Q'''| \leq (1 + c/\sqrt{k})|Q^*_{opt}|$ for some constant c.

Proof. Lemma 3.3.4 follows from Lemma 3.3.2 and Theorem 3.3.3.

To do so, we assume $r = k/c_2$ in Theorem 3.3.3, then $|V_i| \leq k$. Let $Q'''_i = Q''' \cap V_i$ and $Q^*_{opt_i} = Q^*_{opt} \cap V_i$. From Lemma 3.3.2, $|Q'''_i| \leq |Q^*_{opt_i}| + |N(V_i) \cap X|$ for all i. Otherwise, $Q''' \cap N'(V_i)$ can be replaced by $Q^*_{opt_i}$, which contradicts the fact that Q''' is a k-locally optimal subset. Now,

$$|Q'''| \leq |X| + \sum_i |Q'''_i|$$

$$\leq |X| + \sum_i |Q^*_{opt_i}| + \sum_i |N(V_i) \cap X| \quad \text{(See above discussion)}$$

$$\leq \frac{c_1 n}{\sqrt{r}} + |Q^*_{opt}| + \frac{n}{r} c_3 \sqrt{r} \quad \text{(See Theorem 3.3.3)}$$

$$\leq \frac{c_1 n}{\sqrt{r}} + |Q^*_{opt}| + \frac{n}{\sqrt{r}} c_3$$

$$\leq |Q^*_{opt}| + c\frac{n}{\sqrt{r}}$$

$$\leq |Q^*_{opt}| + c\frac{|Q^*_{opt}| + |Q'''|}{\sqrt{r}} = (1 + c/\sqrt{k})|Q^*_{opt}|$$

Thus, $|Q'''| \leq (1 + c/\sqrt{k})|Q^*_{opt}|$, where c is a constant. □

Theorem 3.3.5. $|Q'| \leq (1 + c/\sqrt{k})|Q_{opt}|$ for some constant c.

Proof. Since $Q''' = Q' \setminus I$, $|Q'| = |Q'''| + |I|$. Therefore,

$$|Q'| \leq (1 + c/\sqrt{k})|Q^*_{opt}| + |I| \qquad \text{(By Lemma 3.3.4)}$$

$$= (c/\sqrt{k})|Q^*_{opt}| + |Q^*_{opt}| + |I|$$

$$= (c/\sqrt{k})|Q^*_{opt}| + |Q_{opt}| \qquad \text{(Since } Q^*_{opt} = Q_{opt} \setminus I\text{)}$$

$$\leq (c/\sqrt{k})|Q_{opt}| + |Q_{opt}|$$

$$= (1 + c/\sqrt{k})|Q_{opt}|$$

Thus, $|Q'| \leq (1 + c/\sqrt{k})|Q_{opt}|$ for some constant c. □

Theorem 3.3.6. Algorithm 1 produces $(1 + \epsilon)$-factor approximation result in $O(n^2 m^{2k})$.

Proof. The approximation result follows from Theorem 3.3.5 using $k = O(\epsilon^{-2})$, and the time complexity result follows from Lemma 3.3.1. □

3.4 Conclusion

In this chapter, we studied the (α, P, Q)-covering problem. We proposed a necessary and sufficient condition through which one can ensure the feasibility of the given instance. We proved that the problem is NP-complete for $\alpha \geq 3$. Further, we proposed a local search algorithm that admits a PTAS for the (α, P, Q)-covering problem.

4

Euclidean Dispersion Problem

In this chapter, we study variants of the dispersion problem in Euclidean space. In Chapter 1, we introduced a notion of the dispersion partial sum[1] metric, which captures the idea of dispersion in a more general way. Using the dispersion partial sum as a metric, we define a variant of the dispersion problem, namely the 2-dispersion problem. The 2-dispersion problem is defined formally as follows.

[1]For a facility, the dispersion partial sum is the sum of the distances of the pre-specified number of the closest facilities.

2-Dispersion Problem: *Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points, the non-negative distances between each pair of points $p_i, p_j \in P$, and a positive integer k $(3 \leq k \leq n)$, for each point $p \in P$ and $S \subseteq P$, the 2-dispersion cost of p with respect to S, $cost_2(p, S)$, is defined as the sum of distances from p to the closest point in $S \setminus \{p\}$ and the second closest point in $S \setminus \{p\}$. The 2-dispersion cost of S is defined as $cost_2(S) = \min_{p \in S}\{cost_2(p, S)\}$. The objective of the 2-dispersion problem is to find a subset $S \subseteq P$ of cardinality k such that $cost_2(S)$ is maximized.*

In this chapter, we also study the 1-dispersion problem in Euclidean space. It is defined formally as follows.

1-Dispersion Problem: *Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points, the non-negative distances between each pair of points $p_i, p_j \in P$, and a positive integer k $(2 \leq k \leq n)$, for each point $p \in P$ and $S \subseteq P$, the 1-dispersion cost of p with respect to S, $cost_1(p, S)$, is defined as a distance of p to the closest point in $S \setminus \{p\}$. The 1-dispersion cost of S is defined as $cost_1(S) = \min_{p \in S}\{cost_1(p, S)\}$. The objective of the 1-dispersion problem is to find a subset $S \subseteq P$ of cardinality k such that $cost_1(S)$ is maximized.*

4.0.1 Overview of the Chapter

Goal of the Chapter. (i) Designing a $(2\sqrt{3} + \epsilon)$-factor approximation algorithm for the 2-dispersion problem in $\mathbb{R}^2$, where $\epsilon > 0$, and (ii) developing a common framework for the dispersion problem in Euclidean space using which we improve the approximation factor to $2\sqrt{3}$ for the 2-dispersion problem in $\mathbb{R}^2$, propose an optimal algorithm for the 2-dispersion problem in $\mathbb{R}^1$, and propose a 2-factor approximation result for the 1-dispersion problem in

$\mathbb{R}^2$.

Organization of the Chapter. The remainder of the chapter is organized as follows. In Section 4.1, we propose a $(2\sqrt{3}+\epsilon)$-factor approximation algorithm for the 2-dispersion problem in $\mathbb{R}^2$, where $\epsilon > 0$. In Section 4.2, we propose a common framework for the dispersion problem in Euclidean space. Using the framework, we propose a $2\sqrt{3}$-factor approximation algorithm for the 2-dispersion problem in $\mathbb{R}^2$, a polynomial-time optimal algorithm for the 2-dispersion problem on a line, and a 2-factor approximation algorithm for the 1-dispersion problem in $\mathbb{R}^2$. Finally, we conclude the chapter in Section 4.3.

4.1 $(2\sqrt{3}+\epsilon)$-Factor Approximation Algorithm

In this section, we propose a $(2\sqrt{3}+\epsilon)$-factor approximation algorithm for the 2-dispersion problem, where $\epsilon > 0$. The algorithm is based on a greedy approach. We briefly discuss the algorithm as follows. Let $I = (P, k)$ be an arbitrary instance of the 2-dispersion problem, where $P = \{p_1, p_2, \ldots, p_n\}$ is the set of n points in $\mathbb{R}^2$ and $k \in [3, n]$ is a positive integer. Initially, we choose a subset $S_3 \subseteq P$ of size 3 such that $cost_2(S_3)$ is maximized. Next, we add a point $p \in P$ into S_3 to construct a set S_4, *i.e.*, $S_4 = S_3 \cup \{p\}$, so that $cost_2(S_4)$ is maximized, and continues this process until the construction of the set S_k of size k. The pseudo-code of the algorithm is described in Algorithm 2.

Let $OPT = \{p_1^*, p_2^*, \ldots, p_k^*\}$ be an optimal solution of the 2-dispersion problem for input $I = (P, k)$. For $p \in P$, we define a disk $D[p]$ as follows: $D[p] = \{q \in \mathbb{R}^2 \mid d(p, q) \leq \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}\}$. Accordingly, we define a subset of disk, $D[S]$, for $S \subseteq P$ as $D[S] = \{D[p] \mid p \in S\}$.

Algorithm 2 Euclidean_Dispersion_Algorithm(P, k)

Input: A set $P = \{p_1, p_2, \ldots, p_n\}$ of n points, and a positive integer k $(3 \leq k \leq n)$.
Output: A subset $S_k \subseteq P$ of size k.

1: Compute $\{p_{i_1}, p_{i_2}, p_{i_3}\} \subseteq P$ such that $cost_2(\{p_{i_1}, p_{i_2}, p_{i_3}\})$ is maximized.
2: $S_3 \leftarrow \{p_{i_1}, p_{i_2}, p_{i_3}\}$
3: **for** $(j = 4, 5, \ldots, k)$ **do**
4: Let $p \in P \setminus S_{j-1}$ such that $cost_2(S_{j-1} \cup \{p\})$ is maximized.
5: $S_j \leftarrow S_{j-1} \cup \{p\}$
6: **end for**
7: return (S_k)

Lemma 4.1.1. For any point $p_i \in P$, $|D[p_i] \cap OPT| \leq 2$.

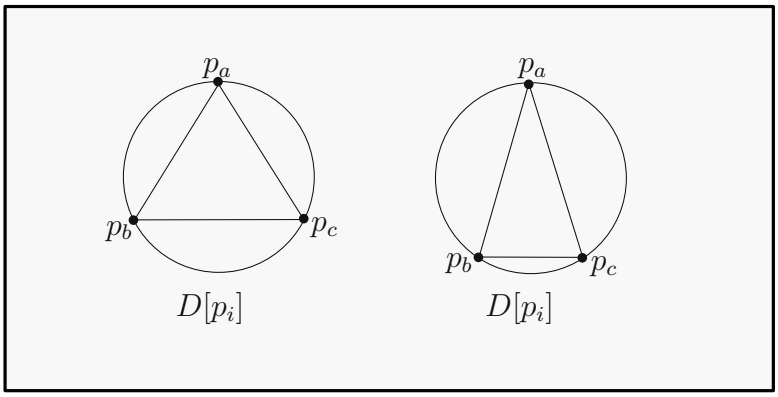


Figure 4.1: *Points* $p_a, p_b, p_c \in D[p_i]$

Proof. On the contrary, assume that there are three points $p_a, p_b, p_c \in D[p_i] \cap OPT$. Let $S = \{p_a, p_b, p_c\}$. Without loss of generality, assume that $cost_2(p_a, S) \leq cost_2(p_b, S)$ and $cost_2(p_a, S) \leq cost_2(p_c, S)$, *i.e.*, $d(p_a, p_b) + d(p_a, p_c) \leq d(p_a, p_b) + d(p_b, p_c)$ and $d(p_a, p_b) + d(p_a, p_c) \leq d(p_a, p_c) + d(p_b, p_c)$, which leads to $d(p_a, p_b) \leq d(p_b, p_c)$ and $d(p_a, p_c) \leq d(p_b, p_c)$. Notice that maximizing $d(p_a, p_b) + d(p_a, p_c)$ results in minimizing $d(p_b, p_c)$(see Figure 4.1). The minimum value of $d(p_b, p_c)$ is $\sqrt{3}\frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$ as both $d(p_a, p_b)$ and $d(p_a, p_c)$ is less than equal to $d(p_b, p_c)$. Therefore, by the packing argument inside a disk, $d(p_a, p_b) + d(p_a, p_c)$ is maximum if p_a, p_b, p_c are on an equilateral triangle and on the boundary of the disk $D[p_i]$.

Then, $cost_2(S) \leq d(p_a, p_b) + d(p_a, p_c) \leq \sqrt{3}\frac{cost_2(OPT)}{2\sqrt{3}+\epsilon} + \sqrt{3}\frac{cost_2(OPT)}{2\sqrt{3}+\epsilon} = 2\sqrt{3}\frac{cost_2(OPT)}{2\sqrt{3}+\epsilon} < cost_2(OPT)$, which leads to a contradiction to the optimal value $cost_2(OPT)$. Therefore, for any $p_i \in P$, $D[p_i]$ contains at most two points in OPT.

□

Consider the set S_i with $i < k$, an i-th size solution in the Algorithm 2. Let $\mathcal{U} = S_i \cap OPT$. Assume that $S_i' = S_i \setminus \mathcal{U}$ and $OPT' = OPT \setminus \mathcal{U}$. Note that for any disk $D[p_\ell^*] \in D[OPT']$, $|D[p_\ell^*] \cap \mathcal{U}| \leq 1$ (by Lemma 4.1.1).

Lemma 4.1.2. For some $D[p_j^*] \in D[OPT']$, $D[p_j^*]$ contains at most one point in S_i, *i.e.*, $|D[p_j^*] \cap S_i| \leq 1$.

Proof. On the contrary, assume that there does not exist any $D[p_j^*] \in D[OPT']$ such that $|D[p_j^*] \cap S_i| \leq 1$, *i.e.*, for each $D[p_v^*] \in D[OPT']$, $|D[p_v^*] \cap S_i| > 1$. Construct a bipartite graph $H(D[OPT'] \cup S_i, \mathcal{E})$ as follows: (i) $D[OPT']$ and S_i are two partite vertex sets, and (ii) $(D[p_j^*], p) \in \mathcal{E}$ if and only if $p \in S_i$ is contained in $D[p_j^*]$(see Figure 4.2).

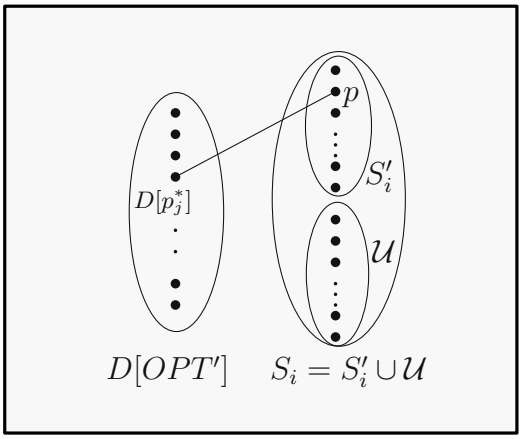


Figure 4.2: $H(D[OPT'] \cup S_i, \mathcal{E})$

Claim 4.1.1. *For a disk $D[p_t^*] \in D[OPT']$, if $|D[p_t^*] \cap \mathcal{U}| = 1$, then any point in $D[p_t^*] \cap S_i'$ is not contained in any disk in $D[OPT'] \setminus \{D[p_t^*]\}$.*

Proof of the Claim. On the contrary assume that a point $p \in D[p_t^*] \cap S'_i$ is contained in a disk $D[p_w^*] \in D[OPT'] \setminus \{D[p_t^*]\}$ (see Figure 4.3). Therefore, $p \in D[p_t^*] \cap D[p_w^*]$ implies $d(p_t^*, p_w^*) \leq 2 \times \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$. Since $|D[p_t^*] \cap \mathcal{U}| = 1$, let $D[p_t^*] \cap \mathcal{U} = \{p_u^*\}$. Now, consider the 2-dispersion cost of p_t^* with respect to OPT, *i.e.*, $cost_2(p_t^*, OPT) \leq d(p_t^*, p_u^*) + d(p_t^*, p_w^*) \leq \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon} + 2 \times \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon} = 3 \times \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon} < cost_2(OPT)$, which is a contradiction to the optimality of OPT (see Figure 4.3). Thus, any point in $D[p_t^*] \cap S'_i$ is not contained in any disk in $D[OPT'] \setminus \{D[p_t^*]\}$, if $|D[p_t^*] \cap \mathcal{U}| = 1$. □

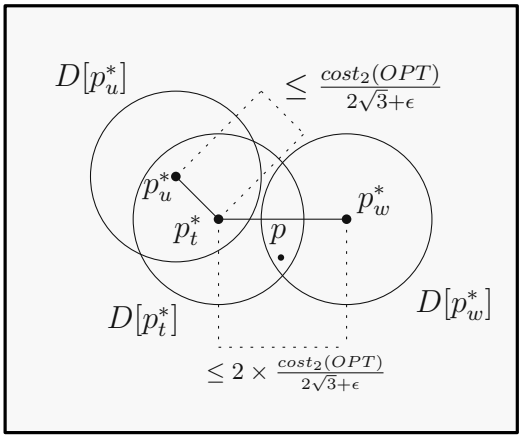


Figure 4.3: *2-dispersion cost of p_t^* with respect to OPT.*

Now, for all $D[p_\ell^*] \in D[OPT']$ that satisfy the condition of Claim 4.1.1, we remove $D[p_\ell^*]$ from $D[OPT']$ to get $D[OPT'']$ and $D[p_\ell^*] \cap S'_i$ from H to get S''_i repeatedly, followed by $\mathcal{U}$ to construct $H' = (D[OPT''], S''_i)$. Since $|D[OPT']| + |\mathcal{U}| = |OPT| = k$, and $|S'_i| + |\mathcal{U}| = |S_i| < k$, therefore $|D[OPT']| > |S'_i|$. During the construction of $H' = (D[OPT''], S''_i)$, the number of vertices removed from the partite set S'_i is at least the number of vertices removed from the partite set $D[OPT']$. Therefore, $|D[OPT'']| > |S''_i|$.

Thus, the lemma follows from the fact that the degree of each vertex in $D[OPT'']$ is at least 2 and the degree of each vertex in S''_i at most 2 in the bipartite graph H', which leads to a contradiction as $|D[OPT'']| > |S''_i|$.

□

Theorem 4.1.3. For any $\epsilon > 0$, Algorithm 2 produces a $(2\sqrt{3}+\epsilon)$-factor approximation result in polynomial time.

Proof. Let $I = (P, k)$ be an arbitrary input instance of the 2-dispersion problem, where $P = \{p_1, p_2, \ldots, p_n\}$ is the set of n points in $\mathbb{R}^2$ and k is a positive integer. Let $S_k = \{p_1, p_2, \ldots, p_k\}$ be the output of Algorithm 2 for instance I. We know that $OPT = \{p_1^*, p_2^*, \ldots, p_k^*\}$ is an optimal solution of the 2-dispersion problem for the instance I.

To prove the theorem, we need to show that $\frac{cost_2(OPT)}{cost_2(S_k)} \leq 2\sqrt{3}+\epsilon$. Here, we use induction to show that $cost_2(S_i) \geq \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$ for each $i = 3, 4, \ldots, k$. Since S_3 is an optimum solution for 3 points (see line number 1 of Algorithm 2), therefore, $cost_2(S_3) \geq cost_2(OPT) \geq \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$ holds. Now, assume that the condition holds for each i such that $3 \leq i < k$. We will prove that the condition, *i.e.*, $cost_2(S_{i+1}) \geq \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$, holds for $(i+1)$ too.

We know by Lemma 4.1.2 that there exists at least one disk $D[p_j^*] \in D[OPT']$ such that $|D[p_j^*] \cap S_i| \leq 1$. Now, consider the case where $|D[p_j^*] \cap S_i| = 1$, then the distance of p_j^* to the second closest point in S_i is greater than $\frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$(see Figure 4.4). Therefore, we can add the point $p_j^* \in OPT$ to the set S_i to construct the set S_{i+1}. Now, consider the case where $|D[p_j^*] \cap S_i| = 0$, then the distance of the point $p_j^* \in OPT$ to any point of S_i is greater than $\frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$. Note that in both cases $cost_2(p_j^*, S_{i+1}) \geq \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$. So, by adding the point p_j^* to the set S_i, we can construct the set S_{i+1} such that $cost_2(S_{i+1}) \geq \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$.

Now, we argue that for any arbitrary point $p \in S_{i+1}$, $cost_2(p, S_{i+1}) \geq \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$. We consider the following two cases: Case (1) p_j^* is not one of the closest points of p in S_{i+1}, and Case (2) p_j^* is one of the closest points of p in S_{i+1}. In the Case (1), $cost_2(p, S_{i+1}) \geq \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$ by the definition of the set S_i. In the Case (2), suppose that p is not contained in the disk $D[p_j^*]$, then $d(p, p_j^*) \geq \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$. This implies $cost_2(p, S_{i+1}) \geq \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$. Now, if p is

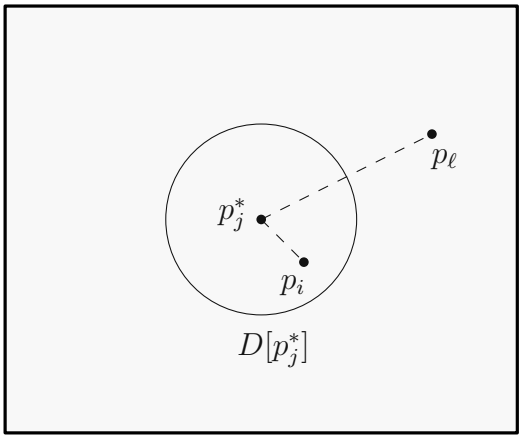


Figure 4.4: *p_ℓ lies outside the disk $D[p_j^*]$*

contained in $D[p_j^*]$, then there exists at least one of the closest points of p that is not contained in $D[p_j^*]$, otherwise it leads to a contradiction to Lemma 4.1.2. Assume that q is one of the nearest points of p that is not contained in $D[p_j^*]$ (see Figure 4.5). Since $d(p,q) \geq \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$, therefore, $cost_2(p, S_{i+1}) \geq \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$. Therefore, by constructing the set $S_{i+1} = S_i \cup \{p_j^*\}$, we ensure that the cost of each point in S_{i+1} is greater than or equal to $\frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$.

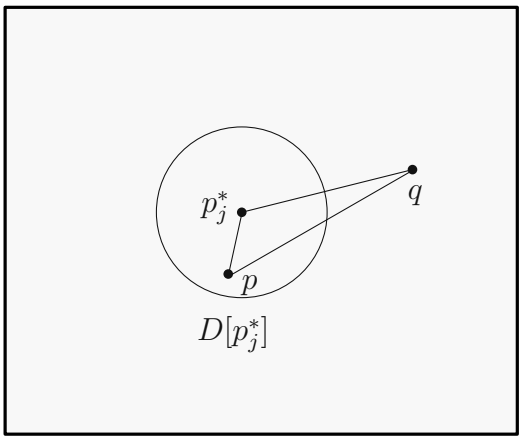


Figure 4.5: *q is not contained in $D[p_j^*]$*

Since our algorithm chooses a point (see line number 4 of Algorithm 2) that maximizes $cost_2(S_{i+1})$, the algorithm will always choose a point in iteration $i+1$ such that $cost_2(S_{i+1}) \geq$

$\frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$.

With the help of Lemma 4.1.1 and Lemma 4.1.2, we conclude that $cost_2(S_{i+1}) \geq \frac{cost_2(OPT)}{2\sqrt{3}+\epsilon}$ and thus the condition also holds for $(i+1)$.

Therefore, for any $\epsilon > 0$, Algorithm 2 produces a $(2\sqrt{3}+\epsilon)$-factor approximation result in polynomial time.

□

4.2 A Common Framework for the Euclidean Dispersion Problem

In this section, we propose a general framework for the dispersion problem. It is a common framework for the 1-dispersion, 2-dispersion problems in $\mathbb{R}^2$ and 1-dispersion / 2-dispersion problems in $\mathbb{R}$. The input of the algorithm is (i) a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points, (ii) an integer γ for the γ-dispersion problem, and (iii) an integer k ($\gamma+1 \leq k \leq n$). In the first line of the algorithm, we set the value of a constant λ. If $\gamma = 2$ and the points are in $\mathbb{R}^2$ (resp. $\mathbb{R}$), then we set $\lambda = 2\sqrt{3}$ (resp. $\lambda = 1$), and if $\gamma = 1$ and the points are in $\mathbb{R}^2$, then we set $\lambda = 2$. We prove that the algorithm is a λ-factor approximation algorithm. We use S_i ($\subseteq P$) to denote a set of points of size i. We start the algorithm with $S_{\gamma+1} \subseteq P$ that contains $\gamma+1$ points as a subset of the solution set. Next, we iteratively add one by one point from P to the set to obtain a final solution set, *i.e.*, if we have a solution set S_i of size i, then we add one more point into S_i to obtain the solution set S_{i+1} of size $i+1$. Let $\alpha = cost_\gamma(S_i)$. Now, we add a point from $P \setminus S_i$ into S_i to get S_{i+1} such that $cost_\gamma(S_{i+1}) \geq \frac{\alpha}{\lambda}$. We stop the iterative method if we have S_k or if adding more points is not possible. We repeat the above process for each distinct $S_{\gamma+1} \subseteq P$ and report the solution for which the γ-*dispersion*

cost value is maximum.

Algorithm 3 Framework_Euclidean_Dispersion(P, k, γ)

Input : A set P of n points, a positive integer γ and an integer k such that $\gamma + 1 \leq k \leq n$.

Output: A subset $S_k \subseteq P$ such that $|S_k| = k$ and $\beta = cost_\gamma(S_k)$.

1: If $\gamma = 2$(resp. $\gamma = 1$), then $\lambda \leftarrow 2\sqrt{3}$ (resp. $\lambda \leftarrow 2$) for points in $\mathbb{R}^2$, and if points are on a line then $\lambda \leftarrow 1$.
2: $\beta \leftarrow 0$ // Initially, $cost_\gamma(S_k) = 0$
3: **for** each subset $S_{\gamma+1} \subseteq P$ consisting of $\gamma + 1$ points **do**
4: Set $\alpha \leftarrow cost_\gamma(S_{\gamma+1})$
5: Set $\rho \leftarrow \alpha/\lambda$
6: **if** $\rho > \beta$ **then**
7: $flag \leftarrow 1$, $i \leftarrow \gamma + 1$
8: **while** $i < k$ and $flag \neq 0$ **do**
9: $flag \leftarrow 0$
10: choose a point $p \in P \setminus S_i$ (if possible) such that $cost_\gamma(S_i \cup \{p\}) \geq \rho$ and $cost_\gamma(p, S_i) = \min_{q \in P \setminus S_i} cost_\gamma(q, S_i)$.
11: **if** such point p exists in step 10 **then**
12: $S_{i+1} \leftarrow S_i \cup \{p\}$
13: $i \leftarrow i + 1$, $flag \leftarrow 1$
14: **end if**
15: **end while**
16: **if** $i = k$ **then**
17: $S_k \leftarrow S_i$ and $\beta \leftarrow \rho$
18: **end if**
19: **end if**
20: **end for**
21: return (S_k, β)

4.2.1 $2\sqrt{3}$-Factor Approximation Result for the 2-Dispersion Problem in $\mathbb{R}^2$

Let $S^* \subseteq P = \{p_1, p_2, \ldots, p_n\}$ be an optimal solution for a given instance (P, k) of the 2-dispersion problem and $S_k \subseteq P$ be a solution returned by greedy Algorithm 3 for the given instance, provided $\gamma = 2$ as an additional input. A point $s_o^* \in S^*$ is said to be a

solution point if $cost_2(S^*)$ is defined by s_o^*, *i.e.*, $cost_2(S^*) = d(s_o^*, s_r^*) + d(s_o^*, s_t^*)$ such that (i) $s_r^*, s_t^* \in S^*$, and (ii) s_r^* and s_t^* are the first and second closest points of s_o^* in S^*, respectively. We call s_r^*, s_t^* supporting points. Let $\alpha = cost_2(S^*)$. Here, we consider the 2-dispersion problem in $\mathbb{R}^2$, so the value of λ is $2\sqrt{3}$ (line number 1 of Algorithm 3).

Lemma 4.2.1. The triangle formed by three points s_o^*, s_r^* and s_t^* does not contain any point in $S^* \setminus \{s_o^*, s_r^*, s_t^*\}$, where s_o^* is the solution point and s_r^*, s_t^* are supporting points.

Proof. Suppose that there exists a point $s_m^* \in S^*$ inside the triangle formed by s_o^*, s_r^*, and s_t^*. Now, if $d(s_o^*, s_r^*) \geq d(s_o^*, s_t^*)$ then $d(s_o^*, s_t^*) + d(s_o^*, s_m^*) < d(s_o^*, s_t^*) + d(s_o^*, s_r^*)$, which contradicts the optimality of $cost_2(S^*)$. A similar argument will also work for $d(s_o^*, s_r^*) < d(s_o^*, s_t^*)$. □

Here, $\rho = \frac{\alpha}{\lambda} = \frac{cost_2(S^*)}{2\sqrt{3}}$. We define the open disk $D(p_i)$ and the closed disk $D[p_i]$ centered at $p_i \in P$ as follows: $D(p_i) = \{p_j \in P \mid d(p_i, p_j) < \rho\}$ and $D[p_i] = \{p_j \in P \mid d(p_i, p_j) \leq \rho\}$. For a subset $S \subseteq P$, let $D(S) = \{D(p) \mid p \in S\}$ and $D[S] = \{D[p] \mid p \in S\}$.

Lemma 4.2.2. For any point $p \in P$, $D(p)$ contains at most two points of the optimal set S^*, *i.e.*, $|D(p) \cap S^*| \leq 2$.

Proof. On the contrary, assume that there are three points $p_a, p_b, p_c \in D(p) \cap S^*$. Using arguments similar to those discussed in the proof of Lemma 4.1.1, $cost_2(\{p_a, p_b, p_c\})$ is the maximum if p_a, p_b, p_c is on the equilateral triangle inside $D(p)$. Therefore, $d(p_a, p_b) = d(p_a, p_c) = d(p_b, p_c)$. Now, $cost_2(\{p_a, p_b, p_c\}) = d(p_a, p_b) + d(p_a, p_c) < \sqrt{3}\rho + \sqrt{3}\rho = 2\sqrt{3}\rho = cost_2(S^*)$. Therefore, $p_a, p_b, p_c \in S^*$ and $cost_2(\{p_a, p_b, p_c\}) < cost_2(S^*)$ lead to a contradiction.

□

Lemma 4.2.3. For any three points $p_a, p_b, p_c \in S^*$, there does not exist any point $p \in P$ such that $p \in D(p_a) \cap D(p_b) \cap D(p_c)$.

Proof. On the contrary, assume that $p \in D(p_a) \cap D(p_b) \cap D(p_c)$. This implies $d(p_a, p) < \rho$, $d(p_b, p) < \rho$ and $d(p_c, p) < \rho$. Therefore, $D(p)$ contains three points p_a, p_b and p_c, which is a contradiction to Lemma 4.2.2. Thus, the lemma.

□

Corollary 4.2.3.1. For any point $p \in P$, if $D'' \subseteq D(S^*)$ is the subset of disks that contains p, then $|D''| \leq 2$.

Proof. The proof follows from Lemma 4.2.3. □

Lemma 4.2.4. For any point $p \in P$, if $D' \subseteq D[S^*]$ is the subset of disks that contains p, then $|D'| \leq 3$ and the point p lies on the boundary of each disk in D'.

Proof. The proof follows from similar arguments in Lemmas 4.1.1 and 4.2.3. □

Let $N \subseteq P$ be a subset of points such that $3 \leq |N| < k$, and $cost_2(N) \geq \rho$. Consider $\mathcal{U} = N \cap S^*$, and let $|\mathcal{U}| = u$. Let $\mathcal{U}' = S^* \setminus \mathcal{U}$ and $N' = N \setminus \mathcal{U}$. Without loss of generality, assume that $\mathcal{U}' = \{p_1^*, p_2^*, \ldots, p_{k-u}^*\}$.

Lemma 4.2.5. For some $D[p_j^*] \in D[\mathcal{U}']$, $D[p_j^*]$ contains at most one point in N, *i.e.*, $|D[p_j^*] \cap N| \leq 1$.

Proof. On the contrary, assume that each $D[p_j^*] \in D[\mathcal{U}']$ contains at least two points in N. Construct a bipartite graph $G(D[\mathcal{U}'] \cup N, \mathcal{E})$ as follows: (i) $D[\mathcal{U}']$ and N are two partite vertex sets, and (ii) $(D[p_j^*], u) \in \mathcal{E}$ if and only if $u \in D[p_j^*]$.

Claim 4.2.1. *For any disk $D[p_m^*] \in D[\mathcal{U}']$, if $|D[p_m^*] \cap \mathcal{U}| = 1$, then any point in $D[p_m^*] \cap N'$ is not contained in any disk in $D[\mathcal{U}'] \setminus \{D[p_m^*]\}$.*

Proof of the Claim. On the contrary, assume that a point $p \in D[p_m^*] \cap N'$ is contained in a disk $D(p_w^*) \in D[\mathcal{U}'] \setminus \{D[p_m^*]\}$ (see Figure 4.6). Therefore, $d(p_m^*, p_w^*) \leq d(p_m^*, p) + d(p, p_w^*) \leq$

$\frac{cost_2(S^*)}{2\sqrt{3}} + \frac{cost_2(S^*)}{2\sqrt{3}}$, which implies $d(p_m^*, p_w^*) \leq 2 \times \frac{cost_2(S^*)}{2\sqrt{3}}$. Since $|D[p_m^*] \cap \mathcal{U}| = 1$, let $D[p_m^*] \cap \mathcal{U} = \{p_y^*\}$. Now, consider the 2-dispersion cost of p_m^* with respect to S^*, *i.e.*, $cost_2(p_m^*, S^*) \leq d(p_m^*, p_y^*) + d(p_m^*, p_w^*) \leq \frac{cost_2(S^*)}{2\sqrt{3}} + 2 \times \frac{cost_2(S^*)}{2\sqrt{3}} = 3 \times \frac{cost_2(S^*)}{2\sqrt{3}} < cost_2(S^*)$, which is a contradiction to the optimality of S^* (see Figure 4.6). Thus, any point in $D[p_m^*] \cap N'$ is not contained in any disk in $D[\mathcal{U}'] \setminus \{D[p_m^*]\}$, if $|D[p_m^*] \cap \mathcal{U}| = 1$.

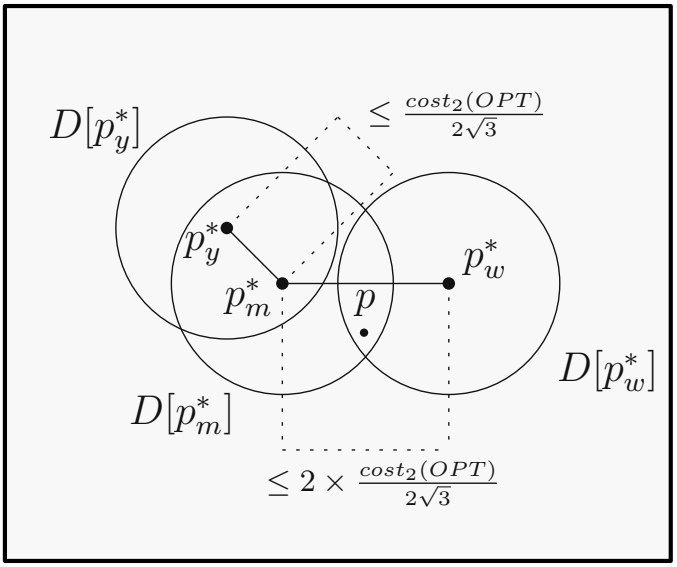


Figure 4.6: 2-*dispersion cost of* p_m^* *with respect to* S^*

Now, for all $D[p_t^*]$ that satisfies the condition in Claim 4.2.1, we remove all such $D[p_t^*]$ and $D[p_t^*] \cap N'$ in G, followed by $\mathcal{U}$ to construct $G' = (D[\mathcal{U}''], N'')$. Since $|D[\mathcal{U}']| + |\mathcal{U}| = |S^*| = k$, and $|N'| + |\mathcal{U}| = |N| < k$, therefore $|D[\mathcal{U}']| > |N'|$. During the construction of $G' = (D[\mathcal{U}''], N'')$, the number of vertices removed from the partite set N' is at least the number of vertices removed from the partite set $D[\mathcal{U}']$ (see Claim (i)). Therefore, $|D[\mathcal{U}'']| > |N''|$.

Thus, the lemma follows from the fact that the degree of each vertex in $D[\mathcal{U}'']$ is at least 2 and the degree of each vertex in N'' is at most 2 in the bipartite graph G', which leads to a contradiction as $|D[\mathcal{U}'']| > |N''|$. Therefore, there exists at least one disk $D[p_j^*] \in D[\mathcal{U}']$

such that $D[p_j^*]$ contains at most one point in N.

□

Theorem 4.2.6. Algorithm 3 produces a $2\sqrt{3}$-factor approximation result for the 2-dispersion problem in $\mathbb{R}^2$.

Proof. Since it is the 2-dispersion problem in $\mathbb{R}^2$, so $\gamma = 2$ and set $\lambda = 2\sqrt{3}$ in line number 1 of Algorithm 3. Now, assume $\alpha = cost_2(S^*)$ and $\rho = \frac{\alpha}{\gamma} = \frac{cost_2(S^*)}{2\sqrt{3}}$, where S^* is an optimum solution. Here, we show that Algorithm 3 returns a solution set S_k of size k such that $cost_2(S_k) \geq \rho = \frac{cost_2(S^*)}{2\sqrt{3}}$. Let s_o^* is the solution point and s_r^* and s_t^* be the supporting points in S^*, *i.e.*, $cost_2(S^*) = d(s_o^*, s_r^*) + d(s_o^*, s_t^*)$. Now, consider the case where $S_3 = \{s_o^*, s_r^*, s_t^*\}$ in line number 3 of Algorithm 3. Our objective is to show that if $S_3 = \{s_o^*, s_r^*, s_t^*\}$ in line number 3 of Algorithm 3, then it computes a solution set S_k of size k such that $cost_2(S_k) \geq \frac{cost_2(S^*)}{2\sqrt{3}}$. Note that any other solution returned by Algorithm 3 has a 2-dispersion cost better than $\frac{cost_2(S^*)}{2\sqrt{3}}$. Therefore, it is sufficient to prove that if $S_3 = \{s_o^*, s_r^*, s_t^*\}$ in line number 3 of Algorithm 3, then the size of S_k (updated) in line number 17 of Algorithm 3 is k as every time Algorithm 3 adds a point (see line number 12) in the set with the property that 2-dispersion cost of the updated set is greater than or equal to $\frac{cost_2(S^*)}{2\sqrt{3}}$. Therefore, we consider $S_3 = \{s_o^*, s_r^*, s_t^*\}$ in line number 3 of Algorithm 3.

We use induction to establish the condition $cost_2(S_i) \geq \rho$ for each $i = 3, 4, \ldots, k$. Since $S_3 = \{s_o^*, s_r^*, s_t^*\}$, therefore $cost_2(S_3) = \alpha > \rho$ holds. Now, assume that the condition $cost_2(S_i) \geq \rho$ holds for each i such that $3 \leq i < k$. We will prove that the condition $cost_2(S_{i+1}) \geq \rho$ holds.

Consider $Y = S_i \cap S^*$ and $\overline{Y} = S^* \setminus Y$. Since $i < k$ and $S_i \subseteq P$ with the condition $cost_2(S_i) \geq \rho$, there exists at least one disk $D[p_j^*]$ centered at a point $p_j^* \in \overline{Y}$ that contains at most one point in S_i (by Lemma 4.2.5). Suppose that $D[p_j^*]$ contains exactly one point

$p_t \in S_i$, then p_t is the first closest point of p_j^* in S_i. Now, by Lemma 4.2.5, we claim that the second closest point p_ℓ of p_j^* in S_i lies outside the disk $D[p_j^*]$ (see Figure 4.7). Since $d(p_j^*, p_\ell) \geq \rho$, therefore $cost_2(p_j^*, S_{i+1}) \geq \rho$. Now, if $D[p_j^*]$ does not contain any point in S_i, then the distance from p_j^* to any point in S_i is greater than or equal to ρ. So, $cost_2(p_j^*, S_{i+1}) \geq \rho$. Thus, we can add the point p_j^* to the set S_i to construct the set S_{i+1}, where $cost_2(p_j^*, S_i) \geq \rho$.

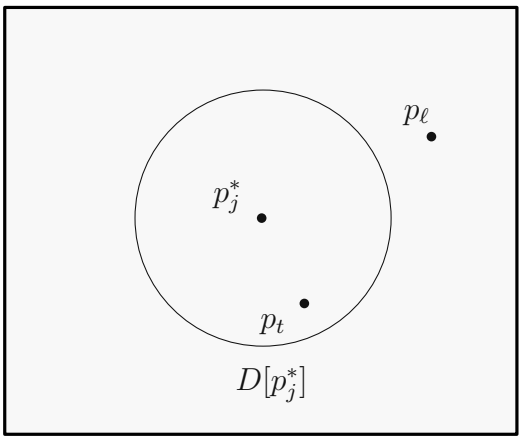


Figure 4.7: *p_ℓ lies outside of the disk $D[p_j^*]$*

Now, we argue for any arbitrary point $p \in S_{i+1}$, $cost_2(p, S_{i+1}) \geq \rho$. We consider the following two cases: Case (1) p_j^* is not one of the closest points to p in S_{i+1}, and the Case (2) p_j^* is one of the closest points to p in S_{i+1}. In the Case (1), $cost_2(p, S_{i+1}) \geq \rho$ by the definition of the set S_i. In the Case (2), suppose that p is not contained in disk $D[p_j^*]$, then $d(p, p_j^*) \geq \rho$. This implies that $cost_2(p, S_{i+1}) \geq \rho$. Now, if p is contained in $D[p_j^*]$, then at least one of the closest points of p is not contained in $D[p_j^*]$, otherwise it leads to a contradiction to Lemma 4.2.5. Assume that q is one of the closest points of p that is not contained in $D[p_j^*]$ (see Figure 4.8). Since $d(p, q) \geq \rho$, therefore, $cost_2(p, S_{i+1}) \geq \rho$. Therefore, by constructing the set $S_{i+1} = S_i \cup \{p_j^*\}$, we ensure that the cost of each point in S_{i+1} is greater than or equal to ρ.

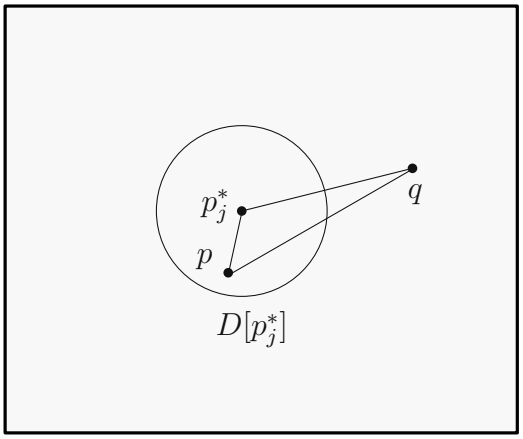


Figure 4.8: *q is not contained in $D[p_j^*]$*

Since there exists at least one point $p_j^* \in \overline{Y}$ such that $cost_2(S_{i+1}) = cost_2(S_i \cup \{p_j^*\}) \geq \rho$, therefore, Algorithm 3 will always choose a point (see line number 10 of Algorithm 3) in iteration $i+1$ such that $cost_2(S_{i+1}) \geq \rho$.

So, we conclude that $cost_2(S_{i+1}) \geq \rho$ and thus the condition also holds for $(i+1)$.

Therefore, Algorithm 3 produces a set S_k of size k such that $cost_2(S_k) \geq \rho$. Since $\rho \geq \frac{cost_2(S^*)}{2\sqrt{3}}$, Algorithm 3 produces $2\sqrt{3}$-factor approximation result for the 2-dispersion problem.

□

4.2.2 2-Dispersion Problem on a Line

In this section, we discuss the 2-dispersion problem on a line L and show that Algorithm 3 produces an optimal solution for it. Let the set $P = \{p_1, p_2, \ldots, p_n\}$ be on a horizontal line arranged from left to right. Let $S_k \subseteq P$ be the solution returned by Algorithm 3 and $S^* \subseteq P$ be an optimal solution. Note that the value of γ is 2 (for the 2-dispersion problem) and the value of λ (line number 1 of Algorithm 3) is 1 in this problem. Let s_o^* be a solution point and s_r^*, s_t^* be supporting points in S^*, *i.e.*, $cost_2(S^*) = d(s_o^*, s_r^*) + d(s_o^*, s_t^*)$. Let $S_3^* = \{s_o^*, s_r^*, s_t^*\}$.

We show that if $S_3 = S_3^*$ in line number 3 of Algorithm 3, then $cost_2(S_k) = cost_2(S^*)$. Let $S^* = \{s_1^*, s_2^*, \ldots s_k^*\}$ be arranged from left to right.

Lemma 4.2.7. If s_o^* is the solution point and s_r^*, s_t^* are supporting points in the optimal solution S^*, then both points s_r^* and s_t^* cannot be on the same side on the line L with respect to s_o^* and three points s_r^*, s_o^*, s_t^* are consecutive on the line L in S^*.

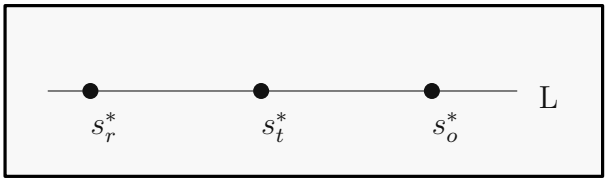


Figure 4.9: *s_r^* and s_t^* on left side of s_o^**

Proof. On the contrary, assume that both s_r^* and s_t^* are on the left side of s_o^*, and s_t^* lies between s_r^* and s_o^* (see Figure 4.9). Now, $d(s_t^*, s_o^*) + d(s_t^*, s_r^*) < d(s_o^*, s_t^*) + d(s_o^*, s_r^*)$ which leads to a contradiction that s_o^* is a solution point, *i.e.*, $cost_2(S^*) = d(s_o^*, s_r^*) + d(s_o^*, s_t^*)$. Now, suppose that s_r^*, s_o^*, s_t^* are not consecutive in S^*. Let s^* be a point in S^* such that either $s^* \in (s_r^*, s_o^*)$ or $s^* \in (s_o^*, s_t^*)$. If $s^* \in (s_r^*, s_o^*)$, then $d(s_o^*, s_r^*) + d(s_o^* + s_t^*) > d(s_o^*, s^*) + d(s_o^* + s_t^*)$, which leads to a contradiction that s_r^* is a supporting point. Similarly, we can show that if $s^* \in (s_o^*, s_t^*)$, then s_t^* is not a supporting point. Thus, s_r^*, s_o^*, s_t^* are consecutive points on the line L in S^*.

□

Lemma 4.2.7 says that if s_o^* is a solution point, then s_{o-1}^* and s_{o+1}^* are supporting points as $s_1^*, s_2^*, \ldots, s_k^*$ are arranged from left to right.

Lemma 4.2.8. Let $S_3 = \{s_o^*, s_{o-1}^*, s_{o+1}^*\}$ and $\alpha = cost_2(S_3)$. Now, if $S_i = S_{i-1} \cup \{p_i\}$ constructed in line number 12 of Algorithm 3, then $cost_2(S_i) = \alpha$.

Proof. We use induction to prove $cost_2(S_i) = \alpha$ for $i = 4, 5 \ldots, k$.

Base Case: Consider the set $S_4 = S_3 \cup \{p_4\}$ constructed in line number 12 of Algorithm 3. If s^*_o is a solution point and s^*_{o-1}, s^*_{o+1} are supporting points and $cost_2(p_4, S_3) \geq \alpha$, therefore $p_4 \notin [s^*_{o-1}, s^*_{o+1}]$ (otherwise one of s^*_{o-1} and s^*_{o+1} will not be a supporting point, for details, see Lemma 4.2.7). This implies p_4 either lies in $[p_1, s^*_{o-1})$ or $(s^*_{o+1}, p_n]$. Assume $p_4 \in (s^*_{o+1}, p_n]$. In Algorithm 3, we choose p_4 such that $cost_2(p_4, S_4) \geq \alpha$ and $cost_2(p_4, S_4) = \min_{q \in P \setminus S_3} cost_2(q, S_4)$ (see line number 10 of Algorithm 3). Therefore, $p_4 \in (s^*_{o+1}, s^*_{o+2}]$. Let $S'_4 = \{s^*_1, s^*_2, \ldots, s^*_{o-2}\} \cup S_4 \cup \{s^*_{o+3}, s^*_{o+4}, \ldots, s^*_k\}$. Suppose $p_4 = s^*_{o+2}$ and we know that $S_3 = S^*_3$ then $S'_4 = S^*$. So, $cost_2(S'_4) = cost_2(S^*) = \alpha$. This implies $cost_2(S_4) = \alpha$. Now assume that $p_4 \in (s^*_{o+1}, s^*_{o+2})$, then we will also show that $cost_2(S'_4) = \alpha$. We calculate $cost_2(p_4, S'_4) = d(p_4, s^*_{o+1}) + d(p_4, s^*_{o+3}) = d(s^*_{o+2}, s^*_{o+1}) + d(s^*_{o+2}, s^*_{o+3}) \geq \alpha$ and $cost_2(s^*_{o+3}, S'_4) = d(s^*_{o+3}, p_4) + d(s^*_{o+3}, s^*_{o+4}) \geq d(s^*_{o+3}, s^*_{o+2}) + d(s^*_{o+3}, s^*_{o+4}) \geq \alpha$ (see Figure 4.10). Thus, if $p_4 \in (s^*_{o+1}, s^*_{o+2})$, then $cost_2(S'_4) = \alpha$. Therefore, if $k \geq 4$, then p_4 exists and $cost_2(S_4) = \alpha$. Similarly, we can prove that if $p_4 \in [p_1, s^*_{o-1})$, then $cost_2(S'_4) = \alpha$, where $S'_4 = \{s^*_1, s^*_2, \ldots, s^*_{o-3}\} \cup S_4 \cup \{s^*_{o+2}, s^*_{o+4}, \ldots, s^*_k\}$. In this case also p_4 exists and $cost_2(S_4) = \alpha$.

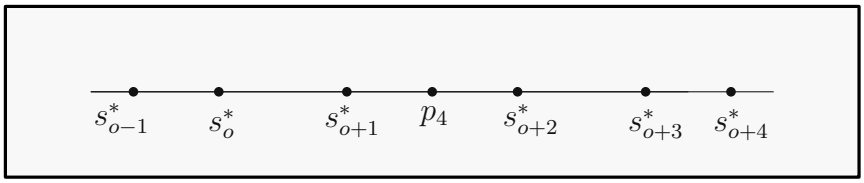


Figure 4.10: *Snippet of S'_4*

Now, assume that $S_i = S_{i-1} \cup \{p_i\}$ for $i < k$ such that $cost_2(S'_i) = \alpha$ and $cost_2(S_i) = \alpha$, where $S'_i = \{s^*_1, s^*_2, \ldots, s^*_u\} \cup S_i \cup \{s^*_v, s^*_{v+1}, \ldots, s^*_k\}$. If $p_i \in (s^*_o, p_n]$, then $s^*_{v-1} \in S^*$ is the left most point in the right of p_i and $u \geq k - (i + k - v + 1) = v - i - 1$ with each point of S_i are on the right side of s^*_u (see Figure 4.11(a)) and if $p_i \in [p_1, s^*_o)$, then $s^*_{u+1} \in S^*$ is the

right most point in the left of p_i, where $v \geq u + i + 1$ (see Figure 4.11(b)).

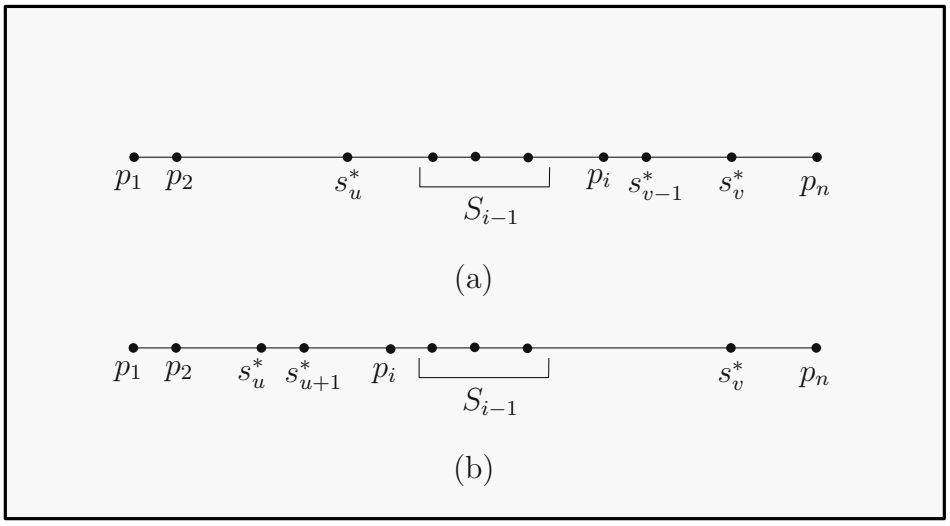


Figure 4.11: *Placement of set $S_{i-1} \cup \{p_i\}$.*

We prove that $cost_2(S_{i+1}) = \alpha$, where $S_{i+1} = S_i \cup \{p_{i+1}\}$. It follows from the fact that the size of S_i is less than k, and the set $\{s^*_1, s^*_2, \ldots, s^*_u\} \cup \{s^*_v, s^*_{v+1}, \ldots, s^*_k\} \neq \phi$ and the similar arguments discussed in the base case. □

Lemma 4.2.9. The running time of Algorithm 3 on the line is $O(n^4)$.

Proof. Since it is the 2-dispersion problem on a line, the algorithm starts by setting $\lambda = 1$ in line number 1 of Algorithm 3, and then compute the solution set for each distinct $S_3 \subseteq P$ independently. Now, for each S_3, the algorithm selects a point iteratively based on the greedy choice (see line number 10 of Algorithm 3). Now, to choose the remaining $(k-3)$ points, the total amortize time taken by the algorithm is $O(n)$. So, the overall time complexity of Algorithm 3 on the line consisting of n points is $O(n^4)$.

□

Theorem 4.2.10. Algorithm 3 produces an optimal solution for the 2-dispersion problem on a line in polynomial time.

Proof. Follows from Lemma 4.2.8 that $cost_2(S_i) = \alpha = cost_2(S_3^*)$ for $3 \leq i \leq k$, where $S_3 = \{s_o^*, s_r^*, s_t^*\}$. Therefore, $cost_2(S_k) = \alpha$. Also, Lemma 4.2.9 says that Algorithm 3 computes S_k in polynomial time. Thus, the theorem. □

4.2.3 1-Dispersion Problem in $\mathbb{R}^2$

In this section, we show the effectiveness of Algorithm 3 by showing 2-factor approximation result for the 1-dispersion problem in $\mathbb{R}^2$. Here, we set $\gamma = 1$ as input along with input P and k. We also set $\lambda = 2$ in line number 1 of the algorithm 3.

Let S^* be an optimal solution for a given instance (P, k) of the 1-dispersion problem and $S_k \subseteq P$ be a solution returned by the greedy Algorithm 3 provided $\gamma = 1$ as an additional input. Let $s_o^* \in S^*$ be a solution point, *i.e.*, $cost_1(S^*) = d(s_o^*, s_r^*)$ such that s_r^* is the closest point to s_o^* in S^*. We call s_r^* the supporting point. Let $\alpha = d(s_o^*, s_r^*)$ and $\rho = \frac{\alpha}{2}$. We define the open disk $D(p_i)$ and the closed disk $D[p_i]$ centered at $p_i \in P$ as follows: $D(p_i) = \{p_j \in P \mid d(p_i, p_j) < \rho\}$ and $D[p_i] = \{p_j \in P \mid d(p_i, p_j) \leq \rho\}$. For $S \subseteq P$, let $D(S) = \{D(p) \mid p \in S\}$ and $D[S] = \{D[p] \mid p \in S\}$.

Lemma 4.2.11. For any point $s \in P$, $D(s)$ contains at most one point of the optimal set S^*.

Proof. On the contrary, assume that $p_a, p_b \in S^*$ such that p_a and p_b are contained in $D(s)$. If two points p_a and p_b are contained in $D(s)$, then $d(p_a, p_b) \leq d(p_a, s) + d(p_b, s) < \frac{\alpha}{2} + \frac{\alpha}{2} = \alpha$, which leads to a contradiction to the optimality of S^*. Thus, the lemma. □

Corollary 4.2.11.1. For any two points $p_a, p_b \in S^*$, there does not exist a point $s \in P$ that is contained in $D(p_a) \cap D(p_b)$.

Proof. On the contrary, assume that s is contained in $D(p_a) \cap D(p_b)$. This implies $d(p_a, s) <$

$\frac{\alpha}{2}$ and $d(p_b, s) < \frac{\alpha}{2}$. Therefore, $D(s)$ contains two points p_a and p_b, which is a contradiction to Lemma 4.2.11. □

Lemma 4.2.12. For any point $s \in P$, if $D' \subseteq D[S^*]$ is the set of disks that contain s, then $|D'| \leq 2$ and the point s lies on the boundary of both disks in D'.

Proof. The proof follows from the similar arguments in Lemma 4.2.11. □

Corollary 4.2.12.1. For any point $s \in P$, if $D'' \subseteq D(S^*)$ is a subset of disks that contains s, then $|D''| \leq 1$.

Proof. Follows from Corollary 4.2.11.1 and Lemma 4.2.12. □

Let $M \subseteq P$ be a subset of points such that $2 \leq |M| < k$, and $cost_1(M) \geq \frac{\alpha}{2}$. Let $\mathcal{V} = M \cap S^*$ and $|\mathcal{V}| = v$. Consider $\mathcal{V}' = S^* \setminus \mathcal{V}$ and $M' = M \setminus \mathcal{V}$. Without loss of generality, assume that $\mathcal{V}' = \{p_1^*, p_2^*, \ldots, p_{k-v}^*\}$.

Lemma 4.2.13. For some $D(p_j^*) \in D(\mathcal{V}')$, $D(p_j^*)$ does not contain any points in M, *i.e.*, $|D(p_j^*) \cap M| = \phi$.

Proof. On the contrary, assume that each $D(p_j^*) \in D(\mathcal{V}')$ contains at least one point in the set M. Now, we construct a bipartite graph $G(D(\mathcal{V}') \cup M', \mathcal{E})$ as follows: (i) $D(\mathcal{V}')$ and M' are two partite vertex sets, and (ii) $(D(p_j^*), u) \in \mathcal{E}$ if and only if $u \in M'$ is contained in $D(p_j^*)$.

According to the assumption, each disk $D(p_j^*) \in D(\mathcal{V}')$ contains at least one point in the set M. Note that disks in $D(\mathcal{V}')$ does not contain a point in $\mathcal{V}$, otherwise a contradiction to Lemma 4.2.11. Therefore, the total degree of the vertices in $D(\mathcal{V}')$ in G is at least $k - v$. On the other hand, the total degree of the vertices in M' in G is at most $|M'| - v$ (see Corollary 4.2.12.1). Since $|M'| < k$, the total degree of the vertices in M' in G is less than $k - v$,

which leads to a contradiction that the total degree of the vertices in $D(\mathcal{V}')$ in G is at least $k - v$. Thus, there exists at least one disk $D(p_j^*) \in D(\mathcal{V}')$ such that $D(p_j^*)$ does not contain any points in M. □

Theorem 4.2.14. Algorithm 3 produces a 2-factor approximation result for the 1-dispersion problem in $\mathbb{R}^2$.

Proof. Since it is the 1-dispersion problem in $\mathbb{R}^2$, so $\gamma = 1$ and set $\lambda = 2$ in line number 1 of Algorithm 3. Now, assume $\alpha = cost_1(S^*)$ and $\rho = \frac{\alpha}{\lambda} = \frac{cost_1(S^*)}{2}$, where S^* is the optimum solution. Here, we show that Algorithm 3 returns a solution set S_k of size k such that $cost_1(S_k) \geq \rho$. More precisely, we show that Algorithm 3 returns a solution S_k of size k such that $cost_1(S_k) \geq \rho$. Let s_o^* be the solution point and s_r^* be the supporting point in S^*. Our objective is to show that if $S_2 = \{s_o^*, s_r^*\}$ in line number 3 of Algorithm 3, then it computes a solution S_k of size k such that $cost_1(S_k) \geq \rho$. Note that any other solution returned by Algorithm 3 has a 1-*dispersion cost* better than $\frac{cost_1(S^*)}{2}$. Therefore, it is sufficient to prove that if $S_2 = \{s_o^*, s_r^*\}$ in line number 3 of Algorithm 3, then the size of S_k (updated) in line number 17 of Algorithm 3 is k as every time Algorithm 3 added a point (see line number 12) into the set with the property that 1-*dispersion cost* of the updated set is greater than or equal to ρ $(= \frac{cost_1(S^*)}{2})$. Therefore, we consider $S_2 = \{s_o^*, s_r^*\}$ in line number 3 of Algorithm 3.

We use induction to establish the condition $cost_1(S_i) \geq \rho$ for each $i = 3, 4, \ldots, k$. Since $S_2 = \{s_o^*, s_r^*\}$, therefore $cost_1(S_2) = \alpha > \rho$ holds. Now, assume that the condition $cost_1(S_i) \geq \rho$ holds for each i such that $3 \leq i < k$. We will prove that condition $cost_1(S_{i+1}) \geq \rho$ also holds for $(i + 1)$.

Consider $Y = S_i \cap S^*$ and $\overline{Y} = S^* \setminus Y$. Since $i < k$ and $S_i \subseteq P$ with the condition $cost_1(S_i) \geq \rho = \frac{\alpha}{2}$, there exists at least one disk, say $D(p_j^*)$ centered at a point $p_j^* \in \overline{Y}$

that does not contain any points in S_i (by Lemma 4.2.13). We will show that $cost_1(S_{i+1}) = cost_1(S_i \cup \{p_j^*\}) \geq \rho$.

Now, if $D(p_j^*)$ does not contain any points in S_i, then the distance of p_j^* to any point in S_i is greater than or equal to ρ (see Figure 4.12). Therefore, $cost_1(p_j^*, S_{i+1}) \geq \rho$. So, we construct $S_{i+1} = S_i \cup \{p_j^*\}$ so that $cost_1(S_{i+1}) \geq \rho$.

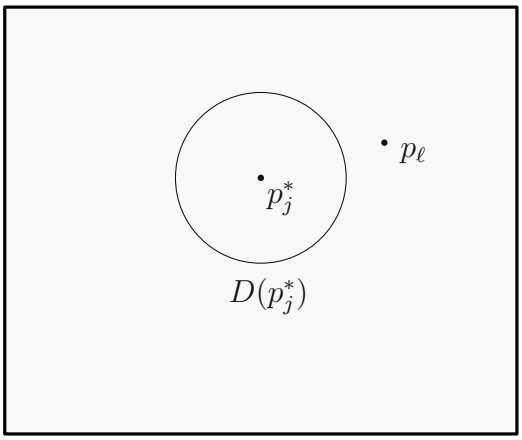


Figure 4.12: *Closest point of p_j^* lies outside of $D(p_j^*)$.*

Now, we argue for any arbitrary point $p \in S_{i+1}$, $cost_1(p, S_{i+1}) \geq \rho$. We consider the following two cases: Case (1) p_j^* is not one of the closest points of p in S_{i+1}, and Case (2) p_j^* is one of the closest points of p in S_{i+1}. In the Case (1), $cost_1(p, S_{i+1}) \geq \rho$ by the definition of the set S_i. In the Case (2), p is not contained in disk $D(p_j^*)$ (by Lemma 4.2.13), thus $d(p, p_j^*) \geq \rho$. This implies that $cost_1(p, S_{i+1}) \geq \rho$. Therefore, by constructing the set $S_{i+1} = S_i \cup \{p_j^*\}$, we ensure that the cost of each point in S_{i+1} is greater than or equal to ρ.

Since $p_j^* \in \overline{Y}$ such that $cost_1(S_{i+1}) = cost_1(S_i \cup \{p_j^*\}) \geq \rho$, therefore Algorithm 3 will always choose a point (see line number 10 of Algorithm 3) in iteration $i+1$ such that $cost_1(S_{i+1}) \geq \rho$.

So, we can conclude that $cost_1(S_{i+1}) \geq \rho$ and thus condition holds for $(i+1)$ too.

Therefore, Algorithm 3 produces a 2-factor approximation result for the 1-dispersion

problem in $\mathbb{R}^2$. □

4.3 Conclusion

In this chapter, we proposed a $(2\sqrt{3}+\epsilon)$-factor approximation algorithm for the 2-dispersion problem in $\mathbb{R}^2$, where $\epsilon > 0$. The best known approximation factor available in the literature is $4\sqrt{3}$ [4]. Next, we proposed a common framework for the dispersion problem. Using the framework, we further improved the approximation factor to $2\sqrt{3}$ for the 2-dispersion problem in $\mathbb{R}^2$. We studied the 2-dispersion problem on a line and proposed a polynomial-time algorithm that returns an optimal solution using the developed framework. Note that for the 2-dispersion problem on a line, one can propose a polynomial-time algorithm that returns an optimal value in relatively low time complexity, but to show the adaptability and flexibility of our proposed framework, we presented an algorithm for the same problem using the developed framework. We also proposed a 2-factor approximation algorithm for the 1-dispersion problem using the proposed common framework to show the effectiveness of the framework.

5

Dispersion Problem in a Metric Space

In this chapter, we study the dispersion problem in a metric space. For the completeness of the chapter, we define the metric space as follows.

Metric Space. A metric space is a pair (X, d), where X is a set of elements and d is a function (metric) from $X \times X$ to $\mathbb{R}$ such that for any $x, y, z \in X$, the following four conditions hold: (i) $d(x, y) \geq 0$, (ii) $d(x, y) = d(y, x)$, (iii) $d(x, y) + d(y, z) \geq d(x, z)$ and (iv) $d(x, y) = 0$ if and only if $x = y$.

In Chapter 1, we introduced the c-dispersion problem using the metric dispersion partial sum[1]. Here, we formally define the c-dispersion problem as follows.

c-Dispersion Problem: *Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points, the non-negative distance $d(p_i, p_j)$ between each pair of points $p_i, p_j \in P$, and a positive integer k $(c+1 \leq k \leq n)$, for each point $p \in P$ and $S \subseteq P$, $cost_c(p, S)$ is defined as the sum of distances from p to the c closest points in $S \setminus \{p\}$. The cost of a subset S of P is defined as $cost_c(S) = \min_{p \in S}\{cost_c(p, S)\}$. The objective of the c-dispersion problem is to find a k size subset S of P such that $cost_c(S)$ is maximized.*

5.0.1 Overview of the Chapter

Goal of the Chapter. To (i) present a polynomial-time algorithm for the c-dispersion problem in a metric space which yields a $2c$-factor approximation result, and (ii) prove that the c-dispersion problem in a metric space parameterized by the solution size k is $W[1]$-hard.

Organization of the Chapter. In Section 5.1, we discuss our proposed algorithm and the analysis of the algorithm in details, we prove the $W[1]$-hardness of the *c-dispersion problem* in a metric space parameterized by the solution size k in Section 5.2, and finally in Section 5.3, we conclude the chapter.

5.1 Algorithm

In this section, we propose an algorithm for the c-dispersion problem in a metric space (X, d). We will show that the proposed algorithm guarantees a $2c$-factor approximation result for the c-dispersion problem in a metric space. Now, we discuss the algorithm as

[1]For a facility, the dispersion partial sum is the sum of the distance of the pre-specified number of the closest facilities.

follows. Let $I = (P, k)$ be an arbitrary instance of the c-dispersion problem in a metric space (X, d), where $P = \{p_1, p_2, \ldots, p_n\}$ is the set of n points and $k \in [c+1, n]$ is a positive integer. It is an iterative algorithm. Initially, we choose a subset $S_{c+1} \subseteq P$ of size $c+1$ such that $cost_c(S_{c+1})$ is maximized. Next, we add a point $p \in P$ to S_{c+1} to construct S_{c+2}, *i.e.*, $S_{c+2} = S_{c+1} \cup \{p\}$, so that $cost_c(S_{c+2})$ is maximized, and continue this process until the construction of S_k of size k. The algorithm is described below.

Algorithm 4 Metric_Dispersion_Algorithm(P, k)

Input: A set $P = \{p_1, p_2, \ldots, p_n\}$ of n points, and a positive integer k $(c+1 \leq k \leq n)$, along with distance function d.
Output: A subset $S_k \subseteq P$ of size k.

1: Compute $S_{c+1} = \{p_{i_1}, p_{i_2}, \ldots, p_{i_c}, p_{i_{c+1}}\} \subseteq P$ so that $cost_c(S_{c+1})$ is maximized by exhaustive search.
2: **for** $(j = c+2, c+3, \ldots, k)$ **do**
3: Let $p \in P \setminus S_{j-1}$ such that $cost_c(S_{j-1} \cup \{p\})$ is maximized.
4: $S_j \leftarrow S_{j-1} \cup \{p\}$
5: **end for**
6: return (S_k)

Let $S^* = \{p_1^*, p_2^*, \ldots, p_k^*\} \subseteq P$ be an optimum solution for the c-dispersion problem for the instance (P, k), *i.e.*, $cost_c(S^*) = \max\limits_{S \subseteq P, |S|=k} \{cost_c(S)\}$. We define the ball $B(p)$ for each point $p \in P$ as follows: $B(p) = \{q \in P | d(p, q) < \frac{cost_c(S^*)}{2c}\}$ and for a subset S of P, let $B(S) = \{B(p) \mid p \in S\}$. See Figure 5.1 for a demonstration of the definition of a ball.

Lemma 5.1.1. For any point $p \in P$, $B(p)$ contains at most c points of the optimal set S^*.

Proof. On the contrary, assume that $B(p)$ contains $c+1$ points of the optimal set S^*. Without loss of generality, assume that $p_1^*, p_2^*, \ldots, p_{c+1}^* (\in S^*)$ are contained in $B(p)$. This implies that $d(p, p_i^*) < \frac{cost_c(S^*)}{2c}$ for each $i = 1, 2, \ldots, c+1$. Since the distance function $d(.,.)$

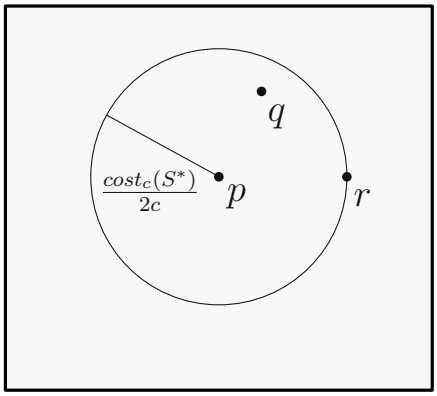


Figure 5.1: *$q \in B(p)$, but $r \notin B(p)$.*

satisfies the triangle inequality, therefore, $d(p_1^*, p_j^*) \leq d(p_1^*, p) + d(p, p_j^*)$, for $j = 2, 3, \ldots, c+1$ (see Figure 5.2). So, this further implies $\sum_{j=2}^{c+1} d(p_1^*, p_j^*) \leq \sum_{j=2}^{c+1} d(p_1^*, p) + \sum_{j=2}^{c+1} d(p, p_j^*) < 2c \times \frac{cost_c(S^*)}{2c} = cost_c(S^*)$. This leads to a contradiction that $p_1^*, p_2^*, \ldots, p_{c+1}^* \in S^*$. □

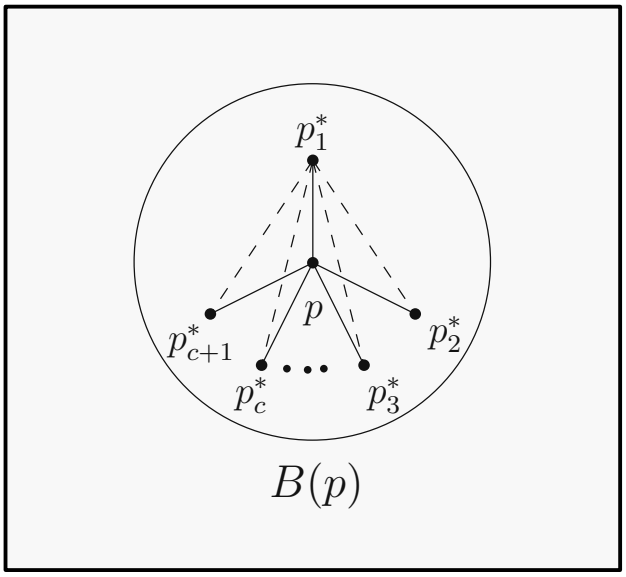


Figure 5.2: *Illustration of Lemma 5.1.1.*

Observation 5.1.2. From Lemma 5.1.1, we observe that for any point $p_i^* \in S^*$, $B(p_i^*)$ contains less than c points of the set $S^* \setminus \{p_i^*\}$.

Lemma 5.1.3. Let $S \subseteq S^*$. If there exists a point $p \in P$ that is included in each ball in $B(S)$, then $|S| \leq c$.

Proof. On the contrary, assume that $|S| > c$. Without loss of generality, assume that $S = \{p_1^*, p_2^*, \ldots, p_c^*, p_{c+1}^*\}$. So, each ball in $B(S) = \{B(p_1^*), B(p_2^*), \ldots, B(p_c^*), B(p_{c+1}^*)\}$ contains p. Since p is contained in $B(p_1^*), B(p_2^*), \ldots, B(p_c^*)$ and $B(p_{c+1}^*)$, therefore, each of $d(p_1^*, p), d(p_2^*, p), \ldots, d(p_c^*, p), d(p_{c+1}^*, p)$ is less than $\frac{cost_c(S^*)}{2c}$. Here, $\{p_1^*, p_2^*, \ldots, p_c^*, p_{c+1}^*\} \subseteq S^*$. So, $B(p)$ contains $c+1$ points, $p_1^*, p_2^*, \ldots, p_c^*, p_{c+1}^*$, of the optimal set S^*, which is a contradiction to Lemma 5.1.1. □

Now, consider a subset $M \subseteq P$ of points such that $c+1 \leq |M| < k$, and $cost_c(M) \geq \frac{cost_c(S^*)}{2c}$. Let $X = M \cap S^*$ and $x = |X|$. Let $\overline{X} = S^* \setminus X$ and $M' = M \setminus X$. Without loss of generality, assume that $\overline{X} = \{p_1^*, p_2^*, \ldots, p_{k-x}^*\}$. Here, $B(\overline{X}) = \{B(p_1^*), B(p_2^*), \ldots, B(p_{k-x}^*)\}$. The following lemma states that there exists a point in the optimal set S^* that is not in the set M, *i.e.*, $p_j^* \in \overline{X}$, such that the ball $B(p_j^*)$ contains less than c points of the set M, *i.e.*, $|B(p_j^*) \cap M| < c$.

Lemma 5.1.4. For some $B(p_j^*) \in B(\overline{X})$, $|B(p_j^*) \cap M| < c$.

Proof. On the contrary, assume that $B(p_j^*) \in B(\overline{X})$ contains at least c points of the set M for each $j \in [1, k-x]$. Note that $B(\overline{X}) = \{B(p_1^*), B(p_2^*), \ldots, B(p_{k-x}^*)\}$. Construct a bipartite graph $G(B(\overline{X}) \cup M, \mathcal{E})$ as follows: (i) $B(\overline{X})$ and M are two partite vertex sets, and (ii) $(B(p_j^*), u) \in \mathcal{E}$ if and only if $u \in M$ is contained in $B(p_j^*) \in B(\overline{X})$ (see Figure 5.3).

Claim 5.1.1. *For $B(p_t^*) \in B(\overline{X})$, if $|B(p_t^*) \cap X| = w$, then less than $(c-w)$ balls in $B(\overline{X}) \setminus \{B(p_t^*)\}$ contain points of $B(p_t^*) \cap M'$, i.e., there exist at most $c-w-1$ balls in $B(\overline{X}) \setminus \{B(p_t^*)\}$ has non-empty intersection with $B(p_t^*) \cap M'$.*

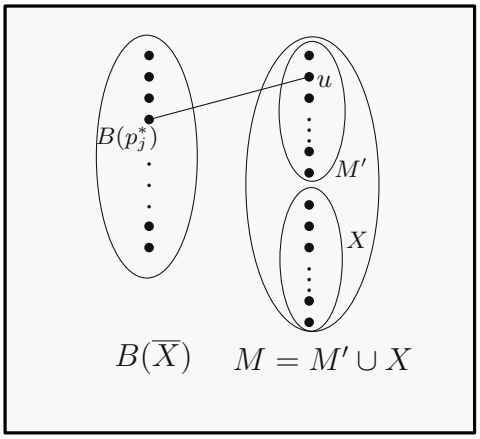


Figure 5.3: $G(B(\overline{X}) \cup M, \mathcal{E})$.

Proof of the Claim. On the contrary, assume that more than $(c-w)$ balls in $B(\overline{X}) \setminus \{B(p_t^*)\}$ has non-empty intersection with $B(p_t^*) \cap M'$. By Observation 5.1.2, $B(p_t^*)$ contains less than c points of the optimal set $S^* \setminus \{p_t^*\}$, and we know that $B(p_t^*)$ contains w points of the set X, so $w < c$. Therefore, $c - w > 0$. We assumed that each $B(p_\ell^*) \in B(\overline{X})$ contains at least c points of M, so $|B(p_t^*) \cap M'| \geq c - w$.

Without loss of generality, assume that each ball in $\{B(p_1^*), B(p_2^*), \ldots, B(p_{c-w}^*)\} \subseteq B(\overline{X}) \setminus \{B(p_t^*)\}$ contains some points of $B(p_t^*) \cap M'$. Since each ball $B(p_\ell^*)$ in $\{B(p_1^*), B(p_2^*), \ldots, B(p_{c-w}^*)\}$ contains some points of $B(p_t^*) \cap M'$, therefore $d(p_\ell^*, p_t^*) \leq 2 \times \frac{cost_c(S^*)}{2c}$ (see the red dashed line in Figure 5.4). Let $B(p_t^*) \cap X = \{p_{x_1}^*, p_{x_2}^*, \ldots, p_{x_w}^*\}$ (all the blue points inside the ball $B(p_t^*)$ in Figure 5.4). Thus, the distance of each (blue) point of $B(p_t^*) \cap X$ to p_t^* is less than or equal to $\frac{cost_c(S^*)}{2c}$ (see the blue dashed line in Figure 5.4).

Note that $\{p_1^*, p_2^*, \ldots, p_{c-w}^*, p_t^*, p_{x_1}^*, p_{x_2}^*, \ldots, p_{x_w}^*\}$ are in S^*, so the c-dispersion cost of p_t^* with respect to S^*, *i.e.*, $cost_c(p_t^*, S^*) \leq \sum_{j=1}^{w} d(p_t^*, p_{x_j}^*) + \sum_{\ell=1}^{c-w} d(p_t^*, p_\ell^*) \leq w \times \frac{cost_c(S^*)}{2c} + (c-w) \times 2 \times \frac{cost_c(S^*)}{2c} = (2c - w) \times \frac{cost_c(S^*)}{2c} < cost_c(S^*)$, which is a contradiction to the optimality. See Figure 5.4 for an illustration of calculating the cost of p_t^* with respect to S^*. Thus, less than $(c-w)$ balls in $B(\overline{X}) \setminus \{B(p_t^*)\}$ contain points of $B(p_t^*) \cap M'$. □

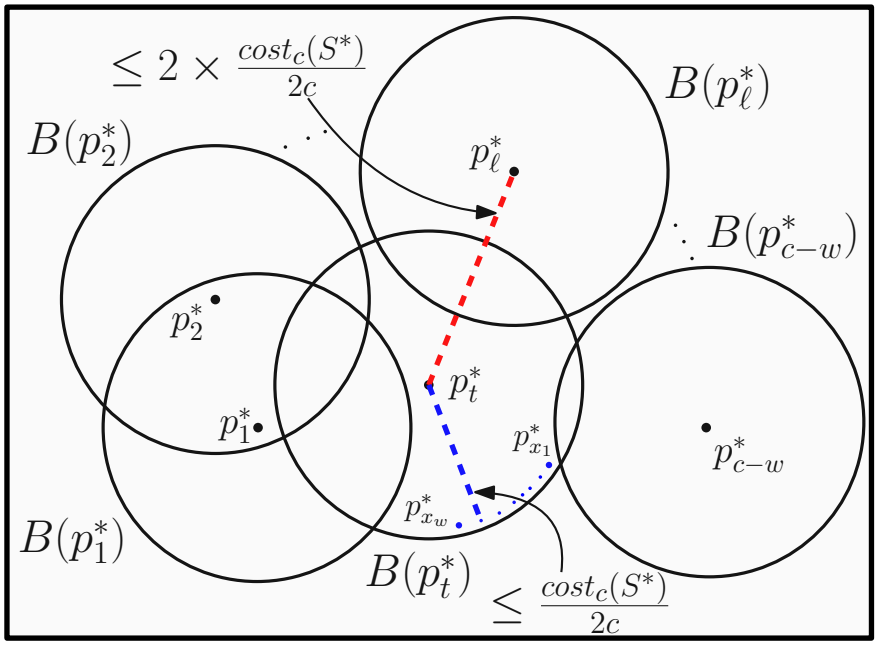


Figure 5.4: *c-dispersion cost of p_t^* with respect to S^**

Now consider a ball $B(p_t^*) \in B(\overline{X})$ that contains w $(< c)$ points from the set X. Since $B(p_t^*)$ contains at least c points (by assumption), it implies $|B(p_t^*) \cap M'| \geq c - w$. By the above claim, the points of $B(p_t^*) \cap M'$ (all the points of M' that are contained in $B(p_t^*)$) are contained in less than $c - w$ balls in $B(\overline{X}) \setminus \{B(p_t^*)\}$. Let the set of balls in $B(\overline{X}) \setminus \{B(p_t^*)\}$ that contains points of $B(p_t^*) \cap M'$ be denoted by $\overline{X_{p_t^*}}$. Consider the connected component $C_{p_t^*}$ in G consisting of $B(p_t^*), \overline{X_{p_t^*}}$ and $B(p_t^*) \cap M'$. We denote the number of vertices of $C_{p_t^*}$ in $B(\overline{X})$ as $\alpha_{p_t^*}$ and the number of vertices of $C_{p_t^*}$ in $B(p_t^*) \cap M'$ as $\beta_{p_t^*}$. Note that $\alpha_{p_t^*} = |\overline{X_{p_t^*}}| + 1$ and $\beta_{p_t^*} \geq c - w$. As $|\overline{X_{p_t^*}}| \leq c - w - 1$, $\alpha_{p_t^*} = |\overline{X_{p_t^*}}| + 1 \leq (c - w - 1) + 1 \leq c - w$. This further implies $\alpha_{p_t^*} \leq c - w \leq \beta_{p_t^*}$. So, $\alpha_{p_t^*} \leq \beta_{p_t^*}$. We remove all such connected components repeatedly (one by one) in G followed by X to construct $G' = (B(\overline{\overline{X}}) \cup M'', \mathcal{E}')$. Note that $B(\overline{\overline{X}}) \subseteq B(\overline{X})$ and $M'' \subseteq M'$. Since $|B(\overline{X})| + |X| = |S^*| = k$ and $|X| + |M'| = |M| < k$, therefore $|B(\overline{X})| > |M'|$. While constructing $G' = (B(\overline{\overline{X}}) \cup M'', \mathcal{E}')$, the number of vertices removed from the partite set $B(\overline{X})$ is at most the number of vertices removed from the

partite set M'. Therefore, $|B(\overline{\overline{X}})| > |M''|$.

Now, we can ensure that the balls in $B(\overline{\overline{X}})$ never contained any points of the set X, otherwise they would have been removed in the construction process of G'. Moreover, by assumption, each ball in $B(\overline{\overline{X}})$ contains at least c points in the set M. Therefore, it implies that each ball in $B(\overline{\overline{X}})$ contains at least c points in the set M'. Furthermore, the construction of G' ensures that each ball in $B(\overline{\overline{X}})$ contains at least c points in the set M'' only.

Thus, the lemma follows from the fact that the degree of each vertex in $B(\overline{\overline{X}})$ is at least c and the degree of each vertex in M'' is at most c (according to Lemma 5.1.3) in the bipartite graph $G' = (B(\overline{\overline{X}}) \cup M'', \mathcal{E}')$, which leads to a contradiction as $|B(\overline{\overline{X}})| > |M''|$. □

Lemma 5.1.5. The running time of Algorithm 4 is $O(\max\{n^{c+1}, n^3\})$.

Proof. In line number 1, the algorithm computes S_{c+1} in a way that maximizes $cost_c(S_{c+1})$. To compute it, the algorithm calculates $cost_c(S_{c+1})$ for each distinct subset $S_{c+1} \subseteq P$ separately. Therefore, the algorithm invests $O\binom{n}{c+1} = O(n^{c+1})$ time to compute S_{c+1} so that $cost_c(S_{c+1})$ is maximized. Now, to choose a point in each iteration, the algorithm takes $O(n^2)$ time. Here, the number of iterations is bounded by $k \leq n$. So, to construct a set S_k of size k from S_{c+1}, the algorithm takes $O(n^3)$ time. Thus, the overall time complexity is $O(\max\{n^{c+1}, n^3\})$. □

Theorem 5.1.6. Algorithm 4 produces $2c$-factor approximation result in polynomial time for the c-dispersion problem.

Proof. Let $I = (P, k)$ be an arbitrary input instance of the c-dispersion problem in a metric space (X, d), where $P = \{p_1, p_2, \ldots, p_n\}$ is the set of n points, and $k \in [c+1, n]$ is a positive integer. Let S_k and S^* be an output of Algorithm 4 and an optimum solution, respectively,

for instance I. To prove the theorem, we show that $\frac{cost_c(S^*)}{cost_c(S_k)} \leq 2c$, *i.e.*, $\frac{cost_c(S^*)}{2c} \leq cost_c(S_k)$. Here, we use induction to show that $cost_c(S_i) \geq \frac{cost_c(S^*)}{2c}$ for each $i = c+1, c+2, \ldots, k$.

Since S_{c+1} is an optimum solution for $c+1$ points (see line number 1 of Algorithm 4), therefore $cost_c(S_{c+1}) \geq cost_c(S^*) \geq \frac{cost_c(S^*)}{2c}$ holds. Now, assume that the condition holds for each i such that $c+1 \leq i < k$. We will prove that the condition holds for $(i+1)$.

Consider $Y = S^* \cap S_i$ and $\overline{Y} = S^* \setminus Y$. Since $i < k$ and $S_i \subseteq P$ such that $cost_c(S_i) \geq \frac{cost_c(S^*)}{2c}$, therefore, there exists at least one ball $B(p_j^*) \in B(\overline{Y})$ that contains less than c points in S_i (see Lemma 5.1.4). So, the distance of p_j^* to one of the c closest points in S_i is greater than or equal to $\frac{cost_c(S^*)}{2c}$. Now, we will show that the algorithm constructs a set S_{i+1} in the $(i+1)$-th iteration such that $cost_c(S_{i+1}) \geq \frac{cost_c(S^*)}{2c}$. We use the following claim to prove the above statement.

Claim 5.1.2. $cost_c(S_i \cup \{p_j^*\}) \geq \frac{cost_c(S^*)}{2c}$.

Proof of the Claim. To prove that $cost_c(S_i \cup \{p_j^*\}) \geq \frac{cost_c(S^*)}{2c}$, we need to ensure that the c-dispersion cost of each point $p \in S_i \cup \{p_j^*\}$ with respect to $S_i \cup \{p_j^*\}$ is greater than or equal to $\frac{cost_c(S^*)}{2c}$, *i.e.*, $cost_c(p, S_i \cup \{p_j^*\}) \geq \frac{cost_c(S^*)}{2c}$. Since the distance of p_j^* to one of the c closest points in S_i is greater than or equal to $\frac{cost_c(S^*)}{2c}$, therefore $cost_c(p_j^*, S_i \cup \{p_j^*\}) \geq \frac{cost_c(S^*)}{2c}$. Now, we will show that the c-dispersion cost of an arbitrary point p $(\neq p_j^*) \in S_i \cup \{p_j^*\}$ with respect to $S_i \cup \{p_j^*\}$ is also greater than or equal to $\frac{cost_c(S^*)}{2c}$. To prove $cost_c(p, S_i \cup \{p_j^*\}) \geq \frac{cost_c(S^*)}{2c}$, we consider the following two cases: Case (1) p_j^* is not one of the c closest points of p in $S_i \cup \{p_j^*\}$, and Case (2) p_j^* is one of the c closest points of p in $S_i \cup \{p_j^*\}$. In Case (1), since p belongs to the set S_i and p_j^* is not one of the c closest points of p in $S_i \cup \{p_j^*\}$, therefore $cost_c(p, S_i \cup \{p_j^*\}) \geq \frac{cost_c(S^*)}{2c}$ (by the definition of S_i). In Case (2), suppose that p is not contained in $B(p_j^*)$, then $d(p, p_j^*) \geq \frac{cost_c(S^*)}{2c}$. This implies $cost_c(p, S_i \cup \{p_j^*\}) \geq \frac{cost_c(S^*)}{2c}$. Now if p is contained in $B(p_j^*)$, then at least one of the c closest points of p in $S_i \cup \{p_j^*\}$ is

not contained in $B(p_j^*)$, otherwise it leads to a contradiction to Lemma 5.1.4. Assume that $q \in S_i \cup \{p_j^*\}$ is one of the c closest points of p that is not contained in $B(p_j^*)$ (see Figure 5.5(a) and 5.5(b)). Thus, the sum of the distances from p to the c closest points in $S_i \cup \{p_j^*\}$ is greater than $d(p, p_j^*) + d(p, q) \geq d(p_j^*, q) \geq \frac{cost_c(S^*)}{2c}$ (in both cases of Figure 5.5(a) and 5.5(b)). So, $cost_c(p, S_i \cup \{p_j^*\}) \geq \frac{cost_c(S^*)}{2c}$. Therefore, $cost_c(S_i \cup \{p_j^*\}) \geq \frac{cost_c(S^*)}{2c}$. □

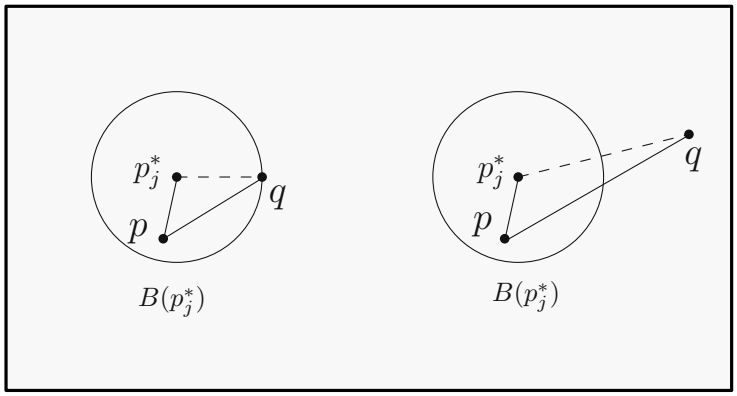


Figure 5.5: *$p \in B(p_j^*)$, and $q \notin B(p_j^*)$.*

By the above claim, we know that there exists a point $p_j^* \in \overline{Y}$, such that $cost_c(S_i \cup \{p_j^*\}) \geq \frac{cost_c(S^*)}{2c}$. Since the algorithm always selects a point (see line number 3 of Algorithm 4) that maximizes $cost_c(S_{i+1})$, therefore it will always select a point in the $(i+1)$-th iteration to construct a set S_{i+1} whose c-dispersion cost is greater than or equal to the c-dispersion cost of $S_i \cup \{p_j^*\}$, *i.e.*, $cost_c(S_{i+1}) \geq cost_c(S_i \cup \{p_j^*\}) \geq \frac{cost_c(S^*)}{2c}$.

With the help of Lemma 5.1.1, Lemma 5.1.3 and Lemma 5.1.4, $cost_c(S_{i+1}) \geq \frac{cost_c(S^*)}{2c}$ and thus the condition holds for $(i+1)$. Also, Lemma 5.1.5 says that Algorithm 4 computes S_k in polynomial time. Therefore, Algorithm 4 produces $2c$-factor approximation result in polynomial time for the c-dispersion problem. □

5.2 A Parameterized Reduction

In this section, we prove that the c-dispersion problem in a metric space (X, d) parameterized by the solution size k is $W[1]$-hard. We show a parameterized reduction from the k-independent set problem parameterized by the solution size k (known to be $W[1]$-hard [43]) to the c-dispersion problem in a metric space (X, d) parameterized by the solution size k. We define a parameterized version of both problems as follows.

k-Independent Set Problem

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

Parameter: k.

Problem: Does there exist an independent set of size k in G ?

c-Dispersion Problem

Instance: A set P of n locations and a positive integer k.

Parameter: k.

Problem: Given a bound $2c$, does there exist $S \subseteq P$ of size k such that $cost_c(S)$ is $2c$?

Theorem 5.2.1. The c-dispersion problem in a metric space (X, d) parameterized by the solution size k is $W[1]$-hard.

Proof. We prove this by giving a parameterized reduction from the k-independent set problem parameterized by the solution size k in simple undirected graphs to the c-dispersion problem in a metric space (X, d) parameterized by the solution size k. Now, we present a method to construct an instance of the c-dispersion problem from an instance of the k-independent set problem in polynomial time.

Let $G = (V, E)$ be an arbitrary instance of the k-independent set problem. Here, $V = \{v_1, v_2, \ldots, v_n\}$. We construct an instance of the c-dispersion problem from the given instance of the k-independent set problem. We use the set V of vertices of G as a set of locations P, *i.e.*, $P = \{p_i \mid v_i \in V\}$ of n points. The distance between the points $p_i, p_j \in P$ as follows: $d(p_i, p_j) = 1$ if $(v_i, v_j) \in E$, and $d(p_i, p_j) = 2$, otherwise. Note that the distance function satisfies the triangle inequality. Thus, the entire process of constructing an instance of the c-dispersion problem in a metric space takes polynomial time.

Claim 5.2.1. *G has an independent set of size k if and only if there exists $S \subseteq P$ of size k, such that $cost_c(S) = 2c$.*

Proof of the Claim. Necessity: Let $I \subseteq V$ be an independent set of G such that $|I| = k$. We construct a set $S \subseteq P$ by selecting k points in P that correspond to the vertices in I, *i.e.*, $S = \{p_i | v_i \in I\}$. Note that $|S| = k$. Since I is an independent set, therefore, by the construction of an instance of the c-dispersion problem from G, the distance between any two points in S is 2. This implies that for each $p \in S$, $cost_c(p, S) = 2c$. Therefore, $cost_c(S) = 2c$.

Sufficiency: Suppose that there exists a $S \subseteq P$ of size k, such that $cost_c(S) = 2c$. Since $cost_c(S) = 2c$, this implies that there exists a point $p \in S$ such that $cost_c(p, S) = 2c$ and for all $q \in S \backslash \{p\}$, $cost_c(q, S) \geq cost_c(p, S) = 2c$. Now, for a point $q \in S \backslash \{p\}$, if $cost_c(q, S) > 2c$, then according to the pigeonhole principle, the distance of q to one of the c closest points in S is greater than 2, which is not possible according to our construction of an instance of the c-dispersion problem. So, for all points $q \in S$, $cost_c(q, S) = 2c$. Now, we can create a set $I \subseteq V$ by selecting the vertices corresponding to each point in S, *i.e.*, $I = \{v_i \mid p_i \in S\}$. Since the distance between each pair of points is 2, therefore there does not exist any edge in I. Thus, $I \subseteq V$ is an independent set of size k. □

Since the k-independent set problem parameterized by the solution size k is $W[1]$-hard, therefore, the reduction in the above claim proves that the c-dispersion problem in a metric space (X, d) parameterized by the solution size k is also $W[1]$-hard. □

5.3 Conclusion

We studied the c-dispersion problem in a metric space. We presented a polynomial-time $2c$-factor approximation algorithm for the c-dispersion problem in a metric space, which is an improvement over the previous best approximation factor $2c^2$ [3]. For $c = 1$, the proposed algorithm produces a 2-factor approximation result, which matches the best known result [70, 76]. We further studied the hardness of the c-dispersion problem in the domain of parameterized complexity and proved that the c-dispersion problem in a metric space is W[1]-hard parameterized by the solution size k.

6

Convex 1-Dispersion Problem

In this chapter, we study a variant of the 1-dispersion problem, where a set of locations is the vertices of a convex polygon. This variant of the 1-dispersion problem is referred to as the convex 1-dispersion problem and is defined as follows.

Convex 1-Dispersion Problem: *Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n vertices of a convex polygon in anti-clockwise order, the Euclidean distance $d(p, q)$ between each pair of vertices $p, q \in P$, the objective of the convex 1-dispersion problem is to find a subset $S \subseteq P$ of size k such that the cost of a subset S, $cost(S) = \min\{d(p, q) \mid p, q \in S\}$, is maximized.*

6.0.1 Overview of the Chapter

Goal of the Chapter. (i) Designing $O(n^3)$ time algorithm for the convex 1-dispersion problem for $k = 4$, and (ii) designing a $\sqrt{3}$ (≈ 1.733)-factor approximation algorithm for the convex 1-dispersion problem.

Organization of the Chapter. The remainder of the Chapter is organized as follows: in Section 6.1, we design an iterative algorithm for the convex 1-dispersion problem for $k = 4$, which produces an optimal solution in $O(n^3)$ time. In Section 6.2, we propose a $\sqrt{3}$-factor approximation result for the convex 1-dispersion problem, and finally conclude the chapter in Section 6.3.

6.1 Convex 1-Dispersion Problem for $k = 4$

In this section, we propose an iterative algorithm for the convex 1-dispersion problem for $k = 4$. To discuss the algorithm in detail, we introduce a few definitions that are required in the design and analysis of the algorithm.

6.1.1 Preliminaries

Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of n vertices of a convex polygon in anti-clockwise order, and for the pair (p_i, p_j) of vertices in P, we define $L_{ij} \subseteq P$ as the subset of vertices to the *left* of the line segment $\overline{p_i p_j}$, *i.e.*, all vertices encountered in counter clockwise order traversing from p_i to p_j (excluding p_i and p_j). Similarly, we define $R_{ij} \subseteq P$ for the pair (p_i, p_j) of vertices as the subset of vertices to the *right* of the line segment $\overline{p_i p_j}$, *i.e.*, all vertices encountered in the clockwise order traversing from p_i to p_j (excluding p_i and p_j). See Figure 6.1(a) for an illustration of both definitions. Now, for the pair (p_i, p_j), we define an *assisting* subset $A_{ij} \subseteq P$ as the set of vertices where the distance of each vertex in A_{ij} from both p_i and p_j is greater than or equal to $d(p_i, p_j)$, *i.e.*, $A_{ij} = \{p_\ell \in P \mid d(p_i, p_\ell) \geq d(p_i, p_j) \wedge d(p_j, p_\ell) \geq d(p_i, p_j)\}$.

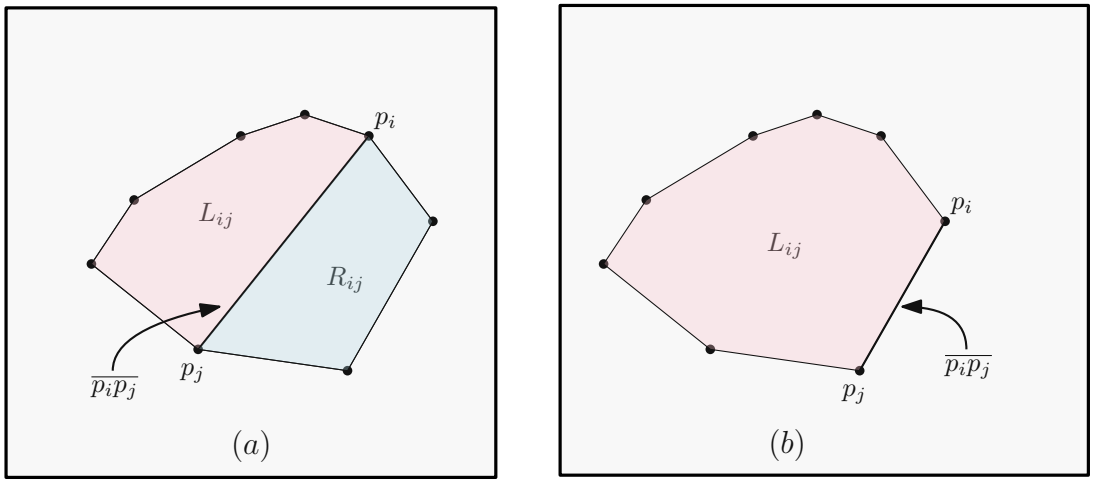


Figure 6.1: *(a) L_{ij}, R_{ij} for the pair $(p_i, p_j) \in P$, and (b) $R_{ij} = \phi$*

For the pair $(p_i, p_j) \in P$, we define a *supporting* subset $S_{ij} \subseteq P$ of four vertices such that $cost(S_{ij}) = d(p_i, p_j)$ and $\{p_i, p_j\} \subseteq S_{ij}$. Note that a supporting subset does not need to exist for each pair of vertices. For example, in the convex polygon in Figure 6.2, there is no supporting subset S_{ij} for the pair (p_i, p_j) as there are no vertices in the polygon whose

distance from both p_i and p_j is greater than or equal to $d(p_i, p_j)$. Furthermore, we define the types of supporting subsets for the pair (p_i, p_j) depending on the positions of both p_i and p_j. If there exists a supporting subset $S_{ij} = \{p_i, p_j, p_k, p_\ell\}$ for the pair (p_i, p_j), then there are three possible orientations for S_{ij} (see Figure 6.3).

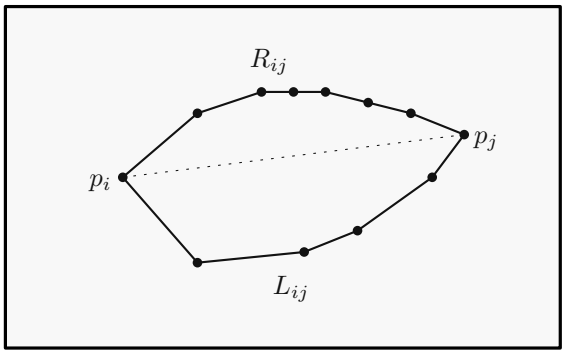


Figure 6.2: *S_{ij} does not exist for the pair (p_i, p_j)*

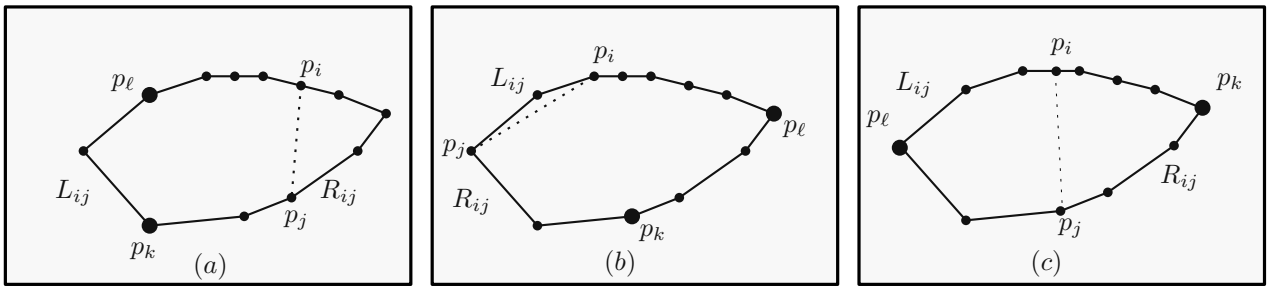


Figure 6.3: *(a) p_k and p_ℓ are in L_{ij}, (b) p_k and p_ℓ are in R_{ij}, and (c) p_k is in L_{ij} and p_ℓ is in R_{ij}*

In Figure 6.3(a) (resp. 6.3(b)), both p_k and p_ℓ are in L_{ij} (resp. R_{ij}). We say such S_{ij} as a *one-sided supporting* subset for the pair (p_i, p_j). On the other hand, in Figure 6.3(c), both p_k and p_ℓ are not together either in L_{ij} or R_{ij}. We say such S_{ij} as a *two-sided supporting* subset for the pair (p_i, p_j).

6.1.2 Algorithm

In this section, we present an iterative algorithm for the convex 1-dispersion problem, where $k = 4$. Now, we discuss the outline of our algorithm, which is as follows. Let $I = (P, 4)$ be an arbitrary instance of the convex 1-dispersion problem. Our algorithm starts by initializing a variable max, which is used to compare costs in each iteration.

We discuss in detail all the steps involved in an iteration for the pair $(p_i, p_j) \in P$. Assume that $d(p_i, p_j) = \rho$. If ρ is less than or equal to the current value of max, then the algorithm does not iterate for the pair (p_i, p_j). Otherwise, the algorithm iterates and decides whether there exists a supporting subset or not. If a supporting subset exists, then max is updated to ρ, else the algorithm proceeds with the other pair of vertices in P. To discuss all steps in an iteration for the pair $(p_i, p_j) \in P$, we assume that $d(p_i, p_j) = \rho$ is greater than the current value of max, *i.e.*, $\rho > max$.

Step (i) First, we traverse P in the clockwise order from p_i in R_{ij} till it finds a vertex p_u that belongs to A_{ij}. Next, we traverse P in counter clockwise order from p_j in R_{ij} until we find a vertex p_v that belongs to A_{ij}. Thereafter, the algorithm compares the distance between p_u and p_v, if $d(p_u, p_v) \geq \rho$, then there exists a one-sided supporting subset $S_{ij} = \{p_i, p_j, p_u, p_v\}$, else we select p_u in $\{p_u, p_v\}$, and further traverse in clockwise order from p_u in R_{uv}, till we either find a vertex p_r such that $d(p_u, p_r) \geq \rho$ or $d(p_v, p_r) \geq \rho$, or finally encounters p_v. If there exists a vertex p_r, then we have a one-sided supporting subset $S_{ij} = \{p_i, p_j, p_r, p_u/p_v\}$; otherwise there does not exist any two vertices in R_{ij} using which we can construct a one-sided supporting subset (see Lemma 6.1.2). In the case of a non-existing one-sided supporting subset using vertices in R_{ij}, we perform the following step.

Step (ii) First, we traverse P in the counter clockwise order from p_i in L_{ij} till we find a vertex p_s that belongs to A_{ij}. Next, we traverse P in clockwise order from p_j in L_{ij}

until we find a vertex p_t that belongs to A_{ij}. Thereafter, the algorithm compares the distance between p_s and p_t, if $d(p_s, p_t) \geq \rho$, then there exists a one-sided supporting subset $S_{ij} = \{p_i, p_j, p_s, p_t\}$, else we select p_s in $\{p_s, p_t\}$, and further traverse in the counter clockwise order from p_s in L_{st}, until we either find a vertex p_w such that $d(p_s, p_w) \geq \rho$ or $d(p_t, p_w) \geq \rho$, or finally encounter p_t. If there exists a vertex p_w, then we have a one-sided supporting subset $S_{ij} = \{p_i, p_j, p_w, p_s/p_t\}$; otherwise there does not exist any two vertices in L_{ij} using which we can construct a one-sided supporting subset (see Lemma 6.1.3). In the case of a non-existing one-sided supporting subset using vertices in L_{ij}, we perform the following step.

Step (iii) So after ensuring that there does not exist any one-sided supporting subset for the pair (p_i, p_j), we try to find a two-sided supporting subset for the pair (p_i, p_j) by checking whether one of $d(p_u, p_s), d(p_u, p_t), d(p_v, p_s), d(p_v, p_t)$ is greater than or equal to ρ or not. If so, then there exists a two-sided supporting $S_{ij} = \{p_i, p_j, p_u/p_v, p_s/p_t\}$; otherwise, there does not exist a two-sided supporting subset S_{ij} for the pair (p_i, p_j) (see Lemma 6.1.4 and Corollary 6.1.4.1).

We iterate the above steps for each pair of vertices in P and return a maximum cost supporting subset. See the pseudocode of the algorithm in Algorithm 5.

Observation 6.1.1. For a pair (p_i, p_j), if $p_u \in R_{ij}$ (resp. $p_t \in L_{ij}$) is the first vertex encountered traversing in clockwise order from p_i (resp. p_j) and $p_v \in R_{ij}$ (resp. $p_s \in L_{ij}$) is the first vertex encountered traversing in counter clockwise order from p_j (resp. p_i), then $R_{uv} \subseteq R_{ij}$ (resp. $L_{st} \subseteq L_{ij}$). See Figure 6.4 for an illustration of the observation.

Consider the following scenario where p_u is the first vertex encountered traversing in clockwise order from p_i and p_v is the first vertex encountered traversing in counter clockwise order from p_j such that both p_u and p_v belong to A_{ij}. Note that $\{p_u, p_v\} \in R_{ij}$. Similarly,

Algorithm 5 Convex 1-Dispersion_Algorithm($P, 4$)

Input: A set $P = \{p_1, p_2, \ldots, p_n\}$ of n vertices of a convex polygon.
Output: An optimal subset $S_{ij} \subseteq P$ and $cost(S_{ij})$.

$max \leftarrow 0$
for each pair $(p_i, p_j) \in P$ **do**
  $d(p_i, p_j) = \rho$
  if $\rho > max$ **then**
    if $R_{ij} \neq \phi$ **then**
      Traverse in the clockwise (resp. counter clockwise) order from p_i (resp. p_j) in R_{ij} till finds a vertex p_u (resp. p_v) that belongs to A_{ij} (if exists).
      if $d(p_u, p_v) \geq \rho$ **then**
        $max \leftarrow \rho$, $S_{ij} = \{p_i, p_j, p_u, p_v\}$, continue;
      else
        Traverse in the clockwise order from p_u in R_{uv} (till we encounter p_v), or finds a vertex $p_r \in R_{uv}$ such that $d(p_u, p_r) \geq \rho$ or $d(p_v, p_r) \geq \rho$ (if exists).
        if $d(p_u, p_r) \geq \rho$ or $d(p_v, p_r) \geq \rho$, **then** $max \leftarrow \rho$, $S_{ij} = \{p_i, p_j, p_r, p_u/p_v\}$, continue;
        end if
      end if
    end if
    if $L_{ij} \neq \phi$ **then**
      Traverse in the counter clockwise (resp. clockwise) order from p_i (resp. p_j) in L_{ij} till finds a vertex p_s (resp. p_t) that belongs to A_{ij} (if exists).
      if $d(p_s, p_t) \geq \rho$ **then**
        $max \leftarrow \rho$, $S_{ij} = \{p_i, p_j, p_s, p_t\}$, continue;
      else
        Traverse in the counter clockwise order from p_s in L_{st} (till we encounter p_t), or finds a vertex $p_w \in L_{st}$ such that $d(p_s, p_w) \geq \rho$ or $d(p_t, p_w) \geq \rho$ (if exists).
        if $d(p_s, p_w) \geq \rho$ or $d(p_t, p_w) \geq \rho$, **then** $max \leftarrow \rho$, $S_{ij} = \{p_i, p_j, p_w, p_s/p_t\}$, continue;
        end if
      end if
    end if
    if ($p_u == p_v$ and $p_s == p_t$) or ($d(p_u, p_s) \geq \rho$ or $d(p_u, p_t) \geq \rho$ or $d(p_v, p_s) \geq \rho$ or $d(p_v, p_t) \geq \rho$) **then**
      $max \leftarrow \rho$, $S_{ij} = \{p_i, p_j, p_u/p_v, p_s/p_t\}$.
    end if
  end if
end for
return S_{ij} and max

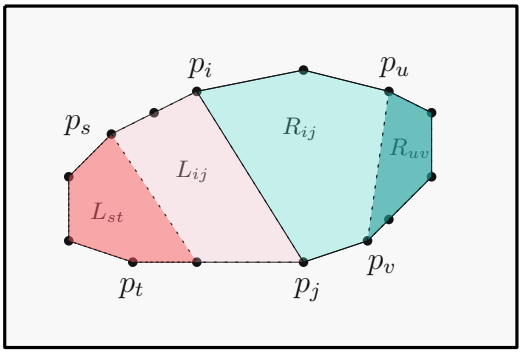


Figure 6.4: *Illustration of Observation 6.1.1*

p_s is the first vertex encountered traversing in counter clockwise order from p_i and p_t is the first vertex encountered traversing in clockwise order from p_j such that both p_s and p_t belong to A_{ij}. Note that $\{p_s, p_t\} \in L_{ij}$.

Lemma 6.1.2. If there exists any one-sided supporting subset S_{ij} such that other vertices apart from p_i and p_j of S_{ij} belong to R_{ij}, then there must exist a one-sided supporting subset S'_{ij} that contains p_u and/or p_v.

Proof. Let S_{ij} be a one-sided supporting subset such that other vertices apart from p_i and p_j of S_{ij} belong to R_{ij}. Let $d(p_i, p_j) = \rho$. Now, we will show that there exists a one-sided supporting subset S'_{ij} that contains p_u and/or p_v. Note that p_u is the first vertex encountered traversing in clockwise order from p_i and p_v is the first vertex encountered traversing in counter clockwise order from p_j such that both p_u and p_v belong to the set A_{ij}. Now, if $d(p_u, p_v) \geq \rho$, then $S'_{ij} = \{p_i, p_j, p_u, p_v\}$ is a one-sided supporting subset containing both p_u and p_v. Now, if $d(p_u, p_v) < \rho$, then $S'_{ij} = \{p_i, p_j, p_u, p_v\}$ is not a supporting subset. By assumption, we know that there exists S_{ij}, so now we will show that there exists a *one-sided supporting* subset S'_{ij} that contains p_u or p_v. Let $S_{ij} = \{p_i, p_j, p_k, p_\ell\}$. Without loss of generality, assume that p_k is encountered before p_ℓ if traversed in clockwise order from p_u. To show that S'_{ij} is a supporting subset containing either p_u or p_v, we

consider a line $\overline{\ell_1}$ passing through both p_i and p_u, and a line $\overline{\ell_2}$ passing through both p_j and p_v, respectively. Since $d(p_u, p_v) < d(p_i, p_j)$, therefore $\overline{\ell_1}$ and $\overline{\ell_2}$ are not parallel to each other (see Figure 6.5). Let q be the intersection point of $\overline{\ell_1}$ and $\overline{\ell_2}$. Since the polygon is convex, therefore, both p_k and p_ℓ are within the triangle $\triangle p_u q p_v$. By assumption, both p_k and p_ℓ belong to S_{ij}, so $d(p_k, p_\ell) \geq \rho$. Now if $\min\{d(p_u, p_k), d(p_v, p_\ell)\} = d(p_u, p_k)$, then $d(p_u, p_\ell) > d(p_k, p_\ell) \geq \rho$, and so $S'_{ij} = \{p_i, p_j, p_u, p_\ell\}$ is a one-sided supporting subset containing p_u. Similarly, if $\min\{d(p_u, p_k), d(p_v, p_\ell)\} = d(p_v, p_\ell)$ then $d(p_v, p_k) > d(p_k, p_\ell) \geq \rho$, and so $S'_{ij} = \{p_i, p_j, p_v, p_k\}$ is a one-sided supporting subset containing p_v. □

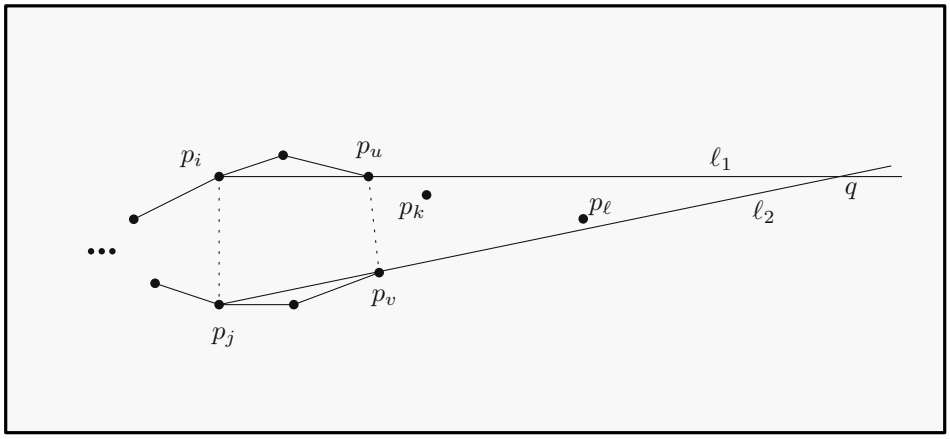


Figure 6.5: *Illustration of Lemma 6.1.2*

Lemma 6.1.3. If there exists any one-sided supporting subset S_{ij} such that other vertices apart from p_i and p_j of S_{ij} belong to L_{ij}, then there must exists a one-sided supporting subset S'_{ij} that contains p_s and/or p_t.

Proof. Similar to the proof of Lemma 6.1.2. □

Lemma 6.1.4. $S_{ij} = \{p_i, p_j, p_u/p_v, p_s/p_t\}$ is a two-sided supporting subset.

Proof. Without loss of generality, assume that $S_{ij} = \{p_i, p_j, p_u, p_s\}$ is not a two-sided supporting subset. Since S_{ij} is not a two-sided supporting subset and $\{p_u, p_s\} \in A_{ij}$, therefore it

implies that $d(p_s, p_u) < d(p_i, p_j)$. Now consider a triangle $\triangle p_s p_i p_u$, since $d(p_i, p_u) \geq d(p_i, p_j)$, $d(p_i, p_s) \geq d(p_i, p_j)$ and $d(p_s, p_u) < d(p_i, p_j)$, therefore $\angle p_s p_i p_u < 60°$. Similarly, we can prove that $\angle p_s p_j p_u < 60°$. Now consider a triangle $\triangle p_i p_u p_j$, since $d(p_i, p_u) \geq d(p_i, p_j)$ and $d(p_j, p_u) \geq d(p_i, p_j)$, therefore $\angle p_i p_u p_j \leq 60°$. Similarly, we can prove that $\angle p_i p_s p_j \leq 60°$. Now, consider the quadrilateral $\diamondsuit p_s p_i p_u p_j$, the sum of the four angles of the quadrilateral $\diamondsuit p_s p_i p_u p_j$ is less than 360°. This is in contradiction to the fact that the sum of the four angles of the quadrilateral is 360°. So, we conclude that $d(p_u, p_s) \geq d(p_i, p_j)$. Thus, $S_{ij} = \{p_i, p_j, p_u, p_s\}$ is a two-sided supporting subset. Using a similar argument, we can prove that $S_{ij} = \{p_i, p_j, p_u, p_t\}$, $S_{ij} = \{p_i, p_j, p_v, p_t\}$ and $S_{ij} = \{p_i, p_j, p_v, p_s\}$ are also a two-sided supporting subset.

□

Now, consider the following scenario where $p_v = p_u$ and $p_t = p_s$ such that both p_v and p_t belong to A_{ij}.

Corollary 6.1.4.1. $S_{ij} = \{p_i, p_j, p_v, p_t\}$ is a two-sided supporting subset.

Proof. Similar to the proof of Lemma 6.1.4. □

Lemma 6.1.5. For any pair (p_i, p_j), if $d(p_i, p_j)$ is greater than the current value of max and there exists a supporting subset S_{ij}, then Algorithm 5 always updates $max = d(p_i, p_j)$.

Proof. Let for the pair (p_i, p_j), $d(p_i, p_j)$ be greater than the current value of max and there is a supporting subset S_{ij}. We know that the algorithm first finds whether there exists a one-sided supporting subset and only if it does not exist, then the algorithm tries to find a two-sided supporting subset.

Now, we will show that the algorithm updates $max = d(p_i, p_j)$ if there exists a (i) one-sided supporting subset and/or (ii) two-sided supporting subset.

For case (i), let S_{ij} be a one-sided supporting subset. Without loss of generality, assume that other vertices apart from p_i and p_j of S_{ij} belong to the set R_{ij}. By Lemma 6.1.2, we know that if there exists any one-sided supporting subset S_{ij}, then there exists a one-sided supporting subset S'_{ij} that contains p_u and/or p_v, where $p_u \in R_{ij}$ is the first vertex encountered traversing in clockwise order from p_i and $p_v \in R_{ij}$ is the first vertex encountered traversing in counter clockwise order from p_j such that $\{p_u, p_v\} \in A_{ij}$. Now, if $d(p_u, p_v) \geq d(p_i, p_j)$, then $S'_{ij} = \{p_i, p_j, p_u, p_v\}$ is a one-sided supporting subset. Thus, $max = d(p_i, p_j)$ is updated in line number 8 of Algorithm 5. Now, if $d(p_u, p_v) < d(p_i, p_j)$, then $S'_{ij} = \{p_i, p_j, p_u, p_v\}$ is not a supporting subset. By assumption, we know that there exists a one-sided supporting subset, so let $S_{ij} = \{p_i, p_j, p_k, p_\ell\}$ be a *one-sided supporting* subset. Note that $\{p_k, p_\ell\}$ belongs to R_{ij}. So, by Lemma 6.1.2, $S'_{ij} = \{p_i, p_j, p_u, p_\ell\}$ or $S'_{ij} = \{p_i, p_j, p_v, p_k\}$ is a one-sided supporting subset. Since the algorithm traverses in clockwise order from p_u, it eventually encounters p_k or p_ℓ. Thus, $max = d(p_i, p_j)$ is updated in line number 11 of Algorithm 5.

For case (ii), since there does not exist a one-sided supporting subset for the pair (p_i, p_j), therefore, the algorithm further tries to find a two-sided supporting subset. Let S_{ij} be a two-sided supporting subset for the pair (p_i, p_j). We know that in the process of finding a one-sided supporting subset for the pair (p_i, p_j), the algorithm traversed in clockwise (resp. counter clockwise) order from p_i (resp. p_j) in R_{ij} and found p_u (resp. p_v). Here, both p_u and p_v belong to the set A_{ij}. Similarly, the algorithm traversed in counter clockwise order (resp. clockwise) from p_i (resp. p_j) in L_{ij} and found p_s (resp. p_t). Here, both p_s and p_t belong to the set A_{ij}. So, by Lemma 6.1.4, $S_{ij} = \{p_i, p_j, p_u/p_v, p_s/p_t\}$ is a two-sided supporting subset. Thus, $max = d(p_i, p_j)$ is updated in line number 26 of Algorithm 5. Moreover, if $p_v = p_u$ and $p_t = p_s$, then $S_{ij} = \{p_i, p_j, p_v, p_t\}$ is a two-sided supporting subset (by Corollary 6.1.4.1), and $max = d(p_i, p_j)$ is updated in line number 26 of Algorithm 5. □

Lemma 6.1.6. The running time of Algorithm 5 is $O(n^3)$.

Proof. Algorithm 5 iterates over all the pair of vertices in P and updates the value of max in each iteration. So, to find a supporting subset in each iteration, the algorithm needs to visit all the vertices in P. Thus, finding a supporting subset in an iteration requires $O(n)$ time. Therefore, the running time of Algorithm 5 is $O(n^3)$. □

Theorem 6.1.7. Algorithm 5 produces an optimal result in $O(n^3)$ time for the convex 1-dispersion problem where $k = 4$.

Proof. Let $S^* = \{p_o^*, p_r^*, p_s^*, p_t^*\} \subseteq P$ be an optimal solution and $S_{ij} = \{p_i, p_j, p_k, p_\ell\} \subseteq P$ be a solution returned by Algorithm 5 for instance $I = (P, 4)$. We know that $cost(S^*) = \max_{\substack{S \subseteq P \\ |S|=4}} \{cost(S)\}$ and let $p_o^* \in S^*$ be a solution vertex and p_r^* be the closet vertex of p_o^* in S^*. So, $cost(S^*) = d(p_o^*, p_r^*)$. Now, we will show that $cost(S_{ij}) = cost(S^*)$.

Let $cost(S_{ij}) \neq cost(S^*)$. So, $cost(S_{ij}) < cost(S^*)$. We know that Algorithm 5 iterated for each pair of vertices in P, so it also iterated for the pair (p_o^*, p_r^*). Since there exists a supporting subset $S_{or} = \{p_o^*, p_r^*, p_s^*, p_t^*\}$, therefore, by Lemma 6.1.5 the value of max must be updated to $d(p_o^*, p_r^*)$ in some iteration of the algorithm. Since our algorithm returns a supporting subset $S_{ij} = \{p_i, p_j, p_k, p_\ell\}$, therefore $d(p_i, p_j)$ must be greater than the value of max in one of the iterations of the algorithm. So, $d(p_i, p_j) > (p_o^*, p_r^*)$. Thus, it implies $cost(S_{ij}) > cost(S^*)$, which is a contradiction that S^* is an optimal solution. Therefore, $cost(S_{ij}) = cost(S^*)$. □

6.2 $\sqrt{3}$-Factor Approximation Result for the Convex 1-Dispersion Problem

In this section, we propose a $\sqrt{3}$-factor approximation algorithm for the convex 1-dispersion problem. The algorithm is based on a greedy approach. We explain the algorithm briefly as follows. Let $I = (P, k)$ be an arbitrary instance of the convex 1-dispersion problem, where $P = \{p_1, p_2, \ldots, p_n\}$ is the set of n vertices of a convex polygon in anti-clockwise order and k is a positive integer. We use $S_i (\subseteq P)$ to denote a set of points of size i. Initially, we start the algorithm $S_2 \subseteq P$ containing 2 points as a subset of the solution set. Let $\alpha = cost(S_2)$. Now, we add a point from $P \setminus S_2$ to S_2 to get S_3 such that $cost(S_3) \geq \frac{\alpha}{\sqrt{3}}$. Next, for the i-th iteration, the algorithm adds a point to the set S_i to obtain S_{i+1}, *i.e.*, if we have a solution set S_i such that $cost(S_i) \geq \frac{\alpha}{\sqrt{3}}$, then the algorithm adds a point to S_i to obtain a set S_{i+1} of size $i+1$ such that $cost(S_{i+1}) \geq \frac{\alpha}{\sqrt{3}}$. We stop this iterative method if we have S_k or no more point addition is possible. We repeat the above process for each distinct $S_2 \subseteq P$ and report the solution for which the cost value is maximum.

Let $S^* \subseteq P$ be an optimal solution and $S_k \subseteq P$ be a solution returned by Algorithm 6 for a given instance (P, k) of the convex 1-dispersion problem. A point $p_o^* \in S^*$ is said to be a solution point if $cost(S^*)$ is defined by p_o^*, *i.e.*, $cost(S^*) = d(p_o^*, p_r^*)$ such that (i) $p_r^* \in S^*$, and (ii) p_r^* is the closest point of p_o^* in S^*. Let $\alpha = cost(S^*)$ and $\rho = \frac{\alpha}{\sqrt{3}}$. For each point $p \in P$, an open disk centered at p is defined as follows: $D(p) = \{q \in \mathbb{R}^2 \mid d(p, q) < \rho\}$.

Lemma 6.2.1. Let $S_2 = \{p_o^*, p_r^*\}$ in line number 2 of Algorithm 6. For each iteration $i \in [2, k-1)$, if the algorithm adds p_i to S_i to construct S_{i+1}, then there exist at least $(k-i-1)$ points in $S^* \setminus S_{i+1}$ such that the distance of each $(k-i-1)$ point from all the points in S_{i+1} is greater than or equal to ρ.

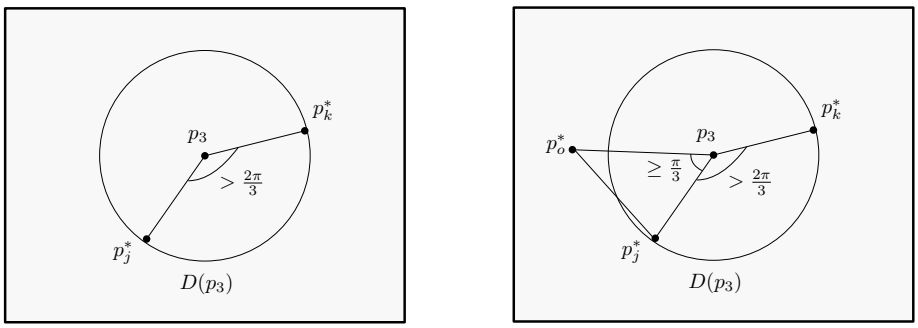
Algorithm 6 Convex 1-Dispersion_Algorithm(P, k)

Input : A set P of n vertices of a convex polygon and an integer k.
Output: A subset $S_k \subseteq P$ such that $|S_k| = k$ and β =cost(S_k).

1: $\beta \leftarrow 0$
2: **for** each subset $S_2 \subseteq P$ consisting of 2 points **do**
3: Set $\alpha \leftarrow cost(S_2)$
4: Set $\rho \leftarrow \frac{\alpha}{\sqrt{3}}$
5: **if** $\rho > \beta$ **then**
6: $flag \leftarrow 1$, $i \leftarrow 2$
7: **while** $i < k$ and $flag \neq 0$ **do**
8: $flag \leftarrow 0$
9: choose a point $p \in P \setminus S_i$ (if possible) such that $cost(S_i \cup \{p\}) \geq \rho$ and $cost(p, S_i) = \min_{q \in P \setminus S_i} cost(q, S_i)$.
10: **if** such point p exists in step 9 **then**
11: $S_{i+1} \leftarrow S_i \cup \{p\}$
12: $i \leftarrow i + 1$, $flag \leftarrow 1$
13: **end if**
14: **end while**
15: **if** $i = k$ **then**
16: $S_k \leftarrow S_i$ and $\beta \leftarrow \rho$
17: **end if**
18: **end if**
19: **end for**
20: return (S_k, β)

Proof. **Base Case.** For $i = 2$. Let p_3 be added to S_2 to construct S_3. Based on the type of points, we classify p_3 in the following two cases: Case (i) $p_3 \in S^*$, and Case (ii) $p_3 \notin S^*$. For Case (i), from the definition of the optimal set S^*, all $k-3$ points of $S^* \setminus S_3$ have distance greater than or equal to ρ with respect to each point in S_3. For Case (ii), if there does not exist at least $(k-2-1) = k-3$ points of $S^* \setminus S_3$ such that their distances from all the points in S_3 is greater than or equal to ρ, then there exist at least two points of $S^* \setminus S_3$ present in $D(p_3)$. Without loss of generality, assume that two points p_j^* and p_k^* are in $D(p_3)$. Since both p_j^* and p_k^* are in $D(p_3)$, therefore $d(p_3, p_j^*) < \rho = \frac{\alpha}{\sqrt{3}}$ and $d(p_3, p_k^*) < \rho = \frac{\alpha}{\sqrt{3}}$. Note that both $p_j^*, p_k^* \in S^*$. So, $d(p_j^*, p_k^*) \geq \alpha$. Thus, $\angle p_j^* p_3 p_k^* > \frac{2\pi}{3}$ (see Figure 6.6(a)). Since p_o^*, p_r^*, p_j^*

and p_k^* are in S^*, the mutual distance between them is greater than or equal to α. The algorithm selected p_3 in iteration 3 of Algorithm 6 and added to the set S_2 to construct S_3, this implies that the minimum distance of p_3 from the points in S_2 must be less than or equal to the minimum distance of p_j^* from the points in S_2. We assumed that p_j^* is in $D(p_3)$. Now, consider a $\triangle p_o^* p_3 p_j^*$, where $d(p_o^*, p_j^*) \geq d(p_o^*, p_3)$ and $d(p_o^*, p_j^*) \geq d(p_3, p_j^*)$, then $\angle p_o^* p_3 p_j^* \geq \frac{\pi}{3}$ (see Figure 6.6(b)). Thus, the internal angle at p_3 is greater than $\frac{2\pi}{3} + \frac{\pi}{3} = \pi$. This leads to the contradiction that the internal angle of a convex polygon is greater than π. Similar arguments will also justify the case for p_r^*. Thus, there exist at least $(k-2-1) = k-3$ points of $S^* \setminus S_3$ such that their distances from all points in S_3 are greater than or equal to ρ.

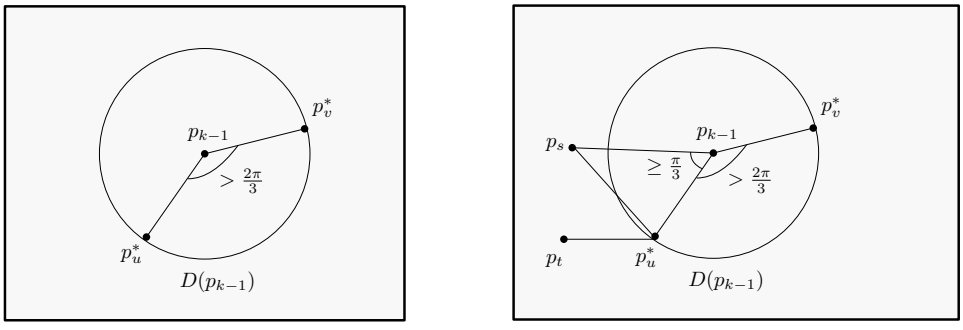


(a) (b)

Figure 6.6: *Illustration of Base Case*

Now, assume that the condition is true for $3 \leq i \leq k-3$. We will show that it is true for $k-2$. Let p_{k-1} be added to S_{k-2} to construct S_{k-1} in $(k-2)$-th iteration. We know by hypothesis that there exist at least $(k-(k-3)-1) = 2$ points of $S^* \setminus S_{k-3}$ after the $(k-3)$-th iteration such that their distances from all the points in S_{k-3} are greater than or equal to ρ. Without loss of generality, assume that p_u^* and p_v^* are two points of $S^* \setminus S_{k-3}$ that satisfy the above condition. Based on the type of points, we classify p_{k-1} in the following two cases: Case (i) $p_{k-1} \in S^*$ and Case (ii) $p_{k-1} \notin S^*$. For Case (i), from

the definition of the optimal set S^* and the induction hypothesis, there exists at least 1 point of $S^* \setminus S_{k-2}$ after the $(k-2)$-th iteration such that their distances from all the points in S_{k-2} are greater than or equal to ρ. For Case (ii), as $p_{k-2} \neq p_u^*$ and $p_{k-2} \neq p_v^*$, and if there does not exist a point of $S^* \setminus S_{k-2}$ after the $(k-2)$-th iteration such that their distances from all the points in S_{k-2} are greater than or equal to ρ, then both p_u^* and p_v^* are in $D(p_{k-1})$. Since both p_u^* and p_v^* are in $D(p_{k-1})$, therefore $d(p_{k-1}, p_u^*) < \rho = \frac{\alpha}{\sqrt{3}}$ and $d(p_{k-1}, p_v^*) < \rho = \frac{\alpha}{\sqrt{3}}$. Note that both $p_u^*, p_v^* \in S^*$. So, $d(p_u^*, p_v^*) \geq \alpha$. Thus, $\angle p_u^* p_{k-1} p_v^* > \frac{2\pi}{3}$ (see Figure 6.7(a)). The algorithm selected p_{k-1} in the $(k-2)$-th iteration and added to the set S_{k-2} to construct S_{k-1}, this implies that the minimum distance of p_{k-1} from the points in S_{k-2} must be less than or equal to the minimum distance of p_u^* from the points in S_{k-2}, *i.e.*, $\min_{q \in S_{k-2}}\{d(p_{k-1}, q)\} \leq \min_{q \in S_{k-2}}\{d(p_u^*, q)\}$. Let $p_s \in S_{k-2}$ be the closest point of p_{k-1} and $p_t \in S_{k-2}$ be the closest point of p_u^*. So, $d(p_s, p_{k-1}) \leq d(p_t, p_u^*) \leq d(p_s, p_u^*)$. Now consider $\triangle p_s p_{k-1} p_u^*$, where $d(p_s, p_u^*) \geq d(p_s, p_{k-1})$ and $d(p_s, p_u^*) \geq d(p_{k-1}, p_u^*)$, then $\angle p_s p_{k-1} p_u^* \geq \frac{\pi}{3}$ (see Figure 6.7(b)). Thus, the internal angle at p_{k-1} is greater than $\frac{2\pi}{3} + \frac{\pi}{3} = \pi$. This leads to the contradiction that the internal angle of a convex polygon is greater than π. Thus, there exists at least $(k - (k-2) - 1) = 1$ point of $S^* \setminus S_{k-2}$ such that their distances from all points in S_{k-2} are greater than or equal to ρ.



(a) (b)

Figure 6.7: *Illustration of Inductive Step*

□

Lemma 6.2.2. The running time of Algorithm is $O(n^4)$.

Proof. Algorithm 6 iterates over all the pair of vertices in P independently, and for each pair, the algorithm try to compute the solution set of size k. The algorithm iterates k times, and in each iteration selects a point based on the greedy choice (see line number 9 of Algorithm 6). Now, for choosing a point in each iteration, algorithm takes $O(n)$ time. Note that $k(\leq n)$. So, to constrict a set S_k of size k, algorithm takes $O(n^2)$ time. Hence, the overall time complexity of Algorithm 6 is $O(n^4)$.

□

Theorem 6.2.3. Algorithm 6 produces a $\sqrt{3}$-factor approximation result for the convex 1-dispersion problem in polynomial time.

Proof. Now, consider the case where $S_2 = \{s_o^*, s_r^*\}$ in line number 2 of Algorithm 6. Our objective is to show that if $S_2 = \{s_o^*, s_r^*\}$ in line number 2 of Algorithm 6, then it computes a solution set S_k of size k such that $cost(S_k) \geq \rho$. Note that any other solution returned by Algorithm 6 has a cost better than ρ. Therefore, it is sufficient to prove that if $S_2 = \{s_o^*, s_r^*\}$ in line number 2 of Algorithm 6, then the size of S_k (updated) in line number 16 of Algorithm 6 is k and $cost(S_k) \geq \rho$, since every time Algorithm 6 added a point (see line number 11) to the set with the property that the cost of the updated set is greater than or equal to ρ. Therefore, we consider $S_2 = \{s_o^*, s_r^*\}$ in line number 2 of Algorithm 6.

By Lemma 6.2.1, we can ensure that if the algorithm starts from $S_2 = \{s_o^*, s_r^*\}$, then after $(k-2)$-th iteration there exists at least $k - (k-2) - 1 = 1$ point of $S^* \setminus S_{k-2}$ such that their distances from all points in S_{k-2} is greater than or equal to ρ. Without loss of generality, assume that $p_t^* \in S^* \setminus S_{k-2}$ is a point that satisfies the above condition.

Observation 6.2.4. $cost(S_{k-2} \cup \{p_t^*\}) \geq \rho$.

Since there exists at least one point $p_t^* \in S^* \setminus S_{k-2}$ such that $cost(S_{k-2} \cup \{p_t^*\}) \geq \rho$, therefore, Algorithm 6 will always choose a point (see line number 9) in $(k-1)$-th iteration such that $cost(S_k) \geq \rho$. So, we can conclude that $cost(S_k) \geq \rho$. Also, Lemma 6.2.2 says that Algorithm 6 computes S_k in $O(n^4)$ time. Therefore, Algorithm 6 produces a $\sqrt{3}$-factor approximation result for the convex 1-dispersion problem in polynomial time.

□

6.3 Conclusion

In this chapter, we studied the convex 1-dispersion problem. We proposed an iterative algorithm that produces an optimal solution for the convex 1-dispersion problem for $k = 4$ in $O(n^3)$ time. To our knowledge, apart from a straightforward $O(n^4)$ time algorithm, so far no other exact algorithm is known for the convex 1-dispersion problem for $k = 4$. The NP-hardness of the convex 1-dispersion problem is unknown, and the problem is open from the point of designing a polynomial-time exact algorithm. Till date, we know that there exists a 2-factor approximation result for the 1-dispersion problem [76], and it cannot be improved further [70]. Thus, applying the same algorithm, we can obtain a 2-factor approximation result for the convex 1-dispersion problem. We designed a $\sqrt{3}$ (≈ 1.733)-factor approximation algorithm for the convex 1-dispersion problem for any value of k, which is a significant improvement over the known result.

7

Conclusion and Future Work

In this chapter, we summarize the work done, highlight the contributions, and suggest directions for possible future work. We have studied a variant of the geometric capacitated set cover problem, namely (α, P, Q)- covering problem, and variants of the dispersion problem, namely, the 2-dispersion problem in $\mathbb{R}^1$ and $\mathbb{R}^2$, the 1-dispersion problem in $\mathbb{R}^2$, the c-dispersion problem in a metric space (X, d), and the convex 1-dispersion problem.

For the (α, P, Q)-covering problem, we established a necessary and sufficient condition that ensures the feasibility of the given instance. We also proposed an algorithm to check the feasibility of an instance of the problem. Moreover, we proved that the (α, P, Q)-covering problem is NP-complete for $\alpha \geq 3$. We proposed a local search algorithm that admits a PTAS. In future work, we would like to design a constant factor approximation algorithm for the (α, P, Q)-covering problem, which has a relatively lower time complexity. We would also like to study the hardness of the (2, P, Q)-covering problem.

For the dispersion problems, we introduced the concept of dispersion partial sum, which generalizes the notion of dispersion. Based on the dispersion partial sum, we defined new variants of the dispersion problem, namely the 1-dispersion problem, the 2-dispersion problem and the c-dispersion problem.

We studied the 2-dispersion problem in $\mathbb{R}^2$, and proposed a polynomial-time algorithm that produces a $(2\sqrt{3}+\epsilon)$-factor approximation result, for any $\epsilon > 0$. Next, we developed a common framework for designing an approximation algorithm for the dispersion problem in Euclidean space. With this common framework, we improved the approximation factor to $2\sqrt{3}$ for the 2-dispersion problem in $\mathbb{R}^2$. We proposed a polynomial-time algorithm, which returns an optimal solution for the 2-dispersion problem when points are placed on a line using the same framework. We also used the same framework to achieve a 2-factor approximation algorithm for the 1-dispersion problem in $\mathbb{R}^2$. The approximation factor results presented in this thesis for both the 1-dispersion problem and the 2-dispersion problem in $\mathbb{R}^2$ are the best to date, so in future work, we would like to improve the approximation factor results for both problems. Moreover, one can think of studying the hardness of the approximation for both problems. We would also like to design an approximation algorithm for the c-dispersion problem in Euclidean space.

Next, we studied the c-dispersion problem in a metric space and presented a polynomial-

time algorithm that yields a $2c$-factor approximation result. For $c = 1$, the proposed algorithm produces a 2-factor approximation result, which matches the best known result [70, 76]. Moreover, unless P=NP, there does not exist a $(2 - \epsilon)$-factor approximation algorithm for the 1-dispersion problem, for any $\epsilon > 0$ [70]. We further showed that the c-dispersion problem in a metric space parameterized by the solution size k is $W[1]$-hard. As a future direction, one can think of investigating the hardness of approximation of the c-dispersion problem for $c > 1$.

Finally, we studied the convex 1-dispersion problem. We proposed an iterative algorithm that produces an optimal solution in $O(n^3)$ time where the objective is to select $k(= 4)$ vertices. We would like to design an efficient algorithm for the convex 1-dispersion problem for any value of k. We also proposed a $\sqrt{3}$-factor approximation algorithm for the convex 1-dispersion problem for any value of k. As a future direction, we would like to improve the approximation factor for the problem.

Publication from the Contents of the Thesis

Papers published/submitted in international journals:

[J1] Pawan K. Mishra, Sangram K. Jena, Gautam K. Das and S. V. Rao, **Capacitated Discrete Unit Disk Cover**, *Discrete Applied Mathematics (DAM), 285: 242-251, 2020.*

[J2] Pawan K. Mishra and Gautam K. Das, **Approximation Algorithms for the Euclidean Dispersion Problems**, (Submitted to Computational Geometry: Theory and Applications (CGTA), Minor Comments Addressed).

[J3] Pawan K. Mishra and Gautam K. Das, **Dispersion Problem in a Metric Space**, (Submitted to Theoretical Computer Science (TCS)).

[J4] Pawan K. Mishra, S. V. Rao and Gautam K. Das, **Dispersion Problem on a Convex Polygon**, (Submitted).

Papers published in international conference proceedings:

[C1] Pawan K. Mishra, Sangrm K. Jena, Gautam K. Das and S. V. Rao, **Capacitated Discrete Unit Disk Cover**, in *In Proceedings of 13th International Conference and Workshop on Algorithms and Computation (WALCOM)*, Lecture Notes in Computer Science, pages 407-418, 2019.

[C2] Pawan K. Mishra and Gautam K. Das, **Approximation Algorithms for the Euclidean Dispersion Problems**, in *In Proceedings of the 33rd Canadian Conference on Computational Geometry (CCCG)*, pages 303-311, 2021.

References

[1] Zoë Abrams, Ashish Goel, and Serge Plotkin. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 424–432, 2004. [Pg.2], [Pg.16]

[2] Toshihiro Akagi, Tetsuya Araki, Takashi Horiyama, Shin-ichi Nakano, Yoshio Okamoto, Yota Otachi, Toshiki Saitoh, Ryuhei Uehara, Takeaki Uno, and Kunihiro Wasa. Exact algorithms for the max-min dispersion problem. In *Proceedings of the 12th International Workshop on Frontiers in Algorithmics*, pages 263–272, 2018. [Pg.25], [Pg.27]

[3] Kazuyuki Amano and Shin-Ichi Nakano. Away from rivals. In *Proceedings of the 30th Canadian Conference on Computational Geometry*, pages 68–71, 2018. [Pg.26], [Pg.86]

[4] Kazuyuki Amano and Shin-Ichi Nakano. An approximation algorithm for the 2-dispersion problem. *IEICE Transactions on Information and Systems*, 103(3):506–508, 2020. [Pg.26], [Pg.72]

[5] Christoph Ambühl, Thomas Erlebach, Matúš Mihalák, and Marc Nunkesser. Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 3–14, 2006. [Pg.23]

[6] Boris Aronov, Esther Ezra, and Micha Sharir. Small-size ϵ-nets for axis-parallel rectangles and boxes. *SIAM Journal on Computing*, 39(7):3248–3282, 2010. [Pg.24]

[7] Manjanna Basappa, Rashmisnata Acharyya, and Gautam K. Das. Unit disk cover problem in 2d. *Journal Discrete Algorithms*, 33:193–201, 2015. [Pg.24]

[8] Christoph Baur and Sándor P Fekete. Approximation of geometric dispersion problems. *Algorithmica*, 30(3):451–470, 2001. [Pg.26]

[9] Mihir Bellare, Shafi Goldwasser, Carsten Lund, and Alexander Russell. Efficient probabilistically checkable proofs and applications to approximations. In *Proceedings of the 25th ACM symposium on Theory of computing*, pages 294–304, 1993. [Pg.21]

[10] Binay Bhattacharya and Qiaosheng Shi. Optimal algorithms for the weighted p-center problems on the real line for small p. In *Proceedings of the 10th Workshop on Algorithms and Data Structures*, pages 529–540, 2007. [Pg.25]

[11] Sayan Bhattacharya, Sreenivas Gollapudi, and Kamesh Munagala. Consideration set generation in commerce search. In *Proceedings of the 20th international conference on World wide web*, pages 317–326, 2011. [Pg.2], [Pg.8]

[12] Therese Biedl and Goos Kant. A better heuristic for orthogonal graph drawings. *Computational Geometry*, 9(3):159–180, 1998. [Pg.34], [Pg.35]

[13] Benjamin Birnbaum and Kenneth J Goldman. An improved analysis for a greedy remote-clique algorithm using factor-revealing lps. *Algorithmica*, 55(1):42–59, 2009. [Pg.26]

[14] Hervé Brönnimann and Michael T Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995. [Pg.22]

[15] Gruia Călinescu, Ion I Mandoiu, Peng-Jun Wan, and Alexander Z Zelikovsky. Selecting forwarding neighbors in wireless ad hoc networks. *Mobile Networks and Applications*, 9(2):101–111, 2004. [Pg.23]

[16] Mihaela Cardei, My T Thai, Yingshu Li, and Weili Wu. Energy-efficient target coverage in wireless sensor networks. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, pages 1976–1984, 2005. [Pg.2], [Pg.16]

[17] Paz Carmi, Matthew J Katz, and Nissan Lev-Tov. Covering points by unit disks of fixed location. In *Proceedings of the 18th International Symposium on Algorithms and Computation*, pages 644–655, 2007. [Pg.23]

[18] Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. Local search for max-sum diversification. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 130–142, 2017. [Pg.2], [Pg.16]

[19] Chi-Kwong Chan and Lee-Ming Cheng. Hiding data in images by simple lsb substitution. *Pattern recognition*, 37(3):469–474, 2004. [Pg.2], [Pg.16]

[20] Timothy M Chan, Elyot Grant, Jochen Könemann, and Malcolm Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In

Proceedings of the 23rd ACM-SIAM symposium on Discrete Algorithms, pages 1576–1585, 2012. [Pg.24]

[21] Barun Chandra and Magnús M Halldórsson. Approximation algorithms for dispersion problems. *Journal of algorithms*, 38(2):438–465, 2001. [Pg.26]

[22] R Chandrasekaran and Andrew Daughety. Location on tree networks: p-centre and n-dispersion problems. *Mathematics of Operations Research*, 6(1):50–57, 1981. [Pg.24]

[23] R Chandrasekaran and Arie Tamir. Polynomially bounded algorithms for locating p-centers on a tree. *Mathematical Programming*, 22(1):304–315, 1982. [Pg.24]

[24] Wang Chi Cheung, Michel X Goemans, and Sam Chiu-wai Wong. Improved algorithms for vertex cover with hard capacities on multigraphs and hypergraphs. In *Proceedings of the 25th ACM-SIAM symposium on Discrete algorithms*, pages 1714–1726, 2014. [Pg.22]

[25] Julia Chuzhoy and Joseph Naor. Covering problems with hard capacities. *SIAM Journal on Computing*, 36(2):498–515, 2006. [Pg.21]

[26] Kenneth L Clarkson and Kasturi Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007. [Pg.24]

[27] Francisco Claude, Gautam K Das, Reza Dorrigiv, Stephane Durocher, Robert Fraser, Alejandro López-Ortiz, Bradford G Nickerson, and Alejandro Salinger. An improved line-separable algorithm for discrete unit disk cover. *Discrete Mathematics, Algorithms and Applications*, 2(01):77–87, 2010. [Pg.23]

[28] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015. [Pg.15]

[29] Gautam K Das, Robert Fraser, Alejandro López-Ortiz, and Bradford G Nickerson. On the discrete unit disk cover problem. *International Journal of Computational Geometry & Applications*, 22(05):407–419, 2012. [Pg.22], [Pg.23]

[30] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of mathematics*, pages 439–485, 2005. [Pg.21]

[31] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the 46th ACM symposium on Theory of computing*, pages 624–633, 2014. [Pg.4], [Pg.21]

[32] Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012. [Pg.15]

[33] Zvi Drezner. *Facility location: a survey of applications and methods*. Springer Series in Operations, 1995. [Pg.1]

[34] Zvi Drezner and Horst W Hamacher. *Facility location: applications and theory*. Springer Science & Business Media, 2004. [Pg.1]

[35] Erhan Erkut. The discrete p-dispersion problem. *European Journal of Operational Research*, 46(1):48–60, 1990. [Pg.26]

[36] Erhan Erkut, Thomas Baptie, and Balder Von Hohenbalken. The discrete p-maxian location problem. *Computers & Operations Research*, 17(1):51–61, 1990. [Pg.24]

[37] Erhan Erkut and Susan Neuman. Analytical models for locating undesirable facilities. *European Journal of Operational Research*, 40(3):275–291, 1989. [Pg.2], [Pg.26]

[38] Erhan Erkut, C ReVelle, and Y Ülküsal. Integer-friendly formulations for the r-separation problem. *European Journal of Operational Research*, 92(2):342–351, 1996. [Pg.2], [Pg.26]

[39] Reza Zanjirani Farahani and Masoud Hekmatfar. *Facility location: concepts, models, algorithms and case studies*. Springer Science & Business Media, 2009. [Pg.1]

[40] Reza Zanjirani Farahani, Maryam SteadieSeifi, and Nasrin Asgari. Multiple criteria facility location problems: A survey. *Applied mathematical modelling*, 34(7):1689–1709, 2010. [Pg.1]

[41] Greg N Federickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, 1987. [Pg.45]

[42] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998. [Pg.21]

[43] Jörg Flum and Martin Grohe. *Parameterized complexity theory*. Springer Science & Business Media, 2006. [Pg.84]

[44] Robert Fraser and Alejandro López-Ortiz. The within-strip discrete unit disk cover problem. *Theoretical Computer Science.*, 674:99–115, 2017. [Pg.23]

[45] Greg N Frederickson. Optimal algorithms for tree partitioning. In *Proceedings of the 2nd ACM-SIAM Symposium on Discrete Algorithms*, pages 168–177, 1991. [Pg.24]

[46] Rajiv Gandhi, Eran Halperin, Samir Khuller, Guy Kortsarz, and Aravind Srinivasan. An improved approximation algorithm for vertex cover with hard capacities. *Journal of Computer and System Sciences*, 72(1):16–33, 2006. [Pg.21]

[47] Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. Dependent rounding and its applications to approximation algorithms. *Journal of the ACM*, 53(3):324–360, 2006. [Pg.21]

[48] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, 1979. [Pg.22], [Pg.33]

[49] AJ Goldman and PM Dearing. Concepts of optimal location for partially noxious facilities. *Bulletin of the Operational Research Society of America*, 23(1):B85, 1975. [Pg.26]

[50] Sreenivas Gollapudi and Aneesh Sharma. An axiomatic approach for result diversification. In *Proceedings of the 18th International conference on World wide web*, pages 381–390, 2009. [Pg.2], [Pg.8]

[51] Sathish Govindarajan, Rajiv Raman, Saurabh Ray, and Aniket Basu Roy. Packing and covering with non-piercing regions. In *Proceedings of the 24th European Symposium on Algorithms*, volume 57, pages 1–17, 2016. [Pg.24]

[52] Sudipto Guha, Refael Hassin, Samir Khuller, and Einat Or. Capacitated vertex covering. *Journal of Algorithms*, 48(1):257–270, 2003. [Pg.21]

[53] Himanshu Gupta, Zongheng Zhou, Samir R Das, and Quinyi Gu. Connected sensor cover: Self-organization of sensor networks for efficient query execution. *IEEE/ACM transactions on networking*, 14(1):55–67, 2006. [Pg.2], [Pg.16]

[54] Refael Hassin, Shlomi Rubinstein, and Arie Tamir. Approximation algorithms for maximum dispersion. *Operations research letters*, 21(3):133–137, 1997. [Pg.26]

[55] David Haussler and Emo Welzl. ϵ-nets and simplex range queries. *Discrete & Computational Geometry*, 2(2):127–151, 1987. [Pg.22]

[56] Dorit S Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and vlsi. *Journal of the ACM*, 32(1):130–136, 1985. [Pg.2], [Pg.16]

[57] Chengbang Huang, Faruck Morcos, Simon P Kanaan, Stefan Wuchty, Danny Z Chen, and Jesus A Izaguirre. Predicting protein-protein interactions from protein domains using a set cover approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(1):78–87, 2007. [Pg.2], [Pg.16]

[58] David S Johnson. Approximation algorithms for combinatorial problems. *Journal of computer and system sciences*, 9(3):256–278, 1974. [Pg.21]

[59] Michael J Kearns. *The computational complexity of machine learning*. MIT press, 1990. [Pg.2]

[60] Jyrki Kivinen, Heikki Mannila, and Esko Ukkonen. Learning hierarchical rule sets. In *Proceedings of the 5th annual workshop on Computational learning theory*, pages 37–44. ACM, 1992. [Pg.2]

[61] Yasuaki Kobayashi, Shin-Ichi Nakano, Kei Uchizawa, Takeaki Uno, Yutaro Yamaguchi, and Katsuhisa Yamanaka. An $o(n^2)$-time algorithm for computing a max-min 3-dispersion on a point set in convex position. *IEICE Transactions on Information and Systems*, 105(3):503–507, 2022. [Pg.27]

[62] Alfred A Kuehn and Michael J Hamburger. A heuristic program for locating warehouses. *Management science*, 9(4):643–666, 1963. [Pg.1]

[63] Ting L Lei and Richard L Church. A unified model for dispersing facilities. *Geographical Analysis*, 45(4):401–418, 2013. [Pg.2], [Pg.26]

[64] Ting L Lei and Richard L Church. On the unified dispersion problem: Efficient formulations and exact algorithms. *European Journal of Operational Research*, 241(3):622–630, 2015. [Pg.2], [Pg.16], [Pg.26]

[65] Nimrod Megiddo and Arie Tamir. An $o(p^2 \log^2 n)$ algorithm for the unweighted p-center problem on the line. Technical report, Technical Report 1981, revised 1991, Department of Statistics, Tel Aviv, 1991. [Pg.25]

[66] I Douglas Moon and Sohail S Chaudhry. An analysis of network location problems with distance constraints. *Management Science*, 30(3):290–307, 1984. [Pg.2], [Pg.26]

[67] Nabil H Mustafa, Rajiv Raman, and Saurabh Ray. Qptas for geometric set-cover problems via optimal separators. *arXiv preprint arXiv:1403.0835*, 2014. [Pg.24]

[68] Nabil H Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010. [Pg.24]

[69] Nabil Hassan Mustafa and Saurabh Ray. Ptas for geometric hitting set problems via local search. In *Proceedings of the 25th Symposium on Computational geometry*, pages 17–22, 2009. [Pg.44]

[70] Sekharipuram S Ravi, Daniel J Rosenkrantz, and Giri Kumar Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299–310, 1994. [Pg.2], [Pg.16], [Pg.25], [Pg.26], [Pg.86], [Pg.105], [Pg.109]

[71] Sanjib Sadhu, Sasanka Roy, Soumen Nandi, Subhas C Nandy, and Suchismita Roy. Efficient algorithm for computing the triangle maximizing the length of its smallest side inside a convex polygon. *International Journal of Foundations of Computer Science*, 31(04):421–443, 2020. [Pg.27]

[72] Barna Saha and Samir Khuller. Set cover revisited: Hypergraph cover with hard capacities. In *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming*, pages 762–773, 2012. [Pg.22]

[73] Douglas R Shier. A min-max theorem for p-center problems on a tree. *Transportation Science*, 11(3):243–252, 1977. [Pg.24]

[74] John F Stollsteimer. A working model for plant numbers and locations. *Journal of Farm Economics*, 45(3):631–645, 1963. [Pg.1]

[75] Marcin Sydow. Approximation guarantees for max sum and max min facility dispersion with parameterised triangle inequality and applications in result diversification. *Mathematica Applicanda*, 42(2):241–257, 2014. [Pg.2], [Pg.16]

[76] Arie Tamir. Obnoxious facility location on graphs. *SIAM Journal on Discrete Mathematics*, 4(4):550–567, 1991. [Pg.25], [Pg.26], [Pg.86], [Pg.105], [Pg.109]

[77] Leslie G. Valiant. Universality considerations in VLSI circuits. *IEEE Transactions on Computers*, 100(2):135–140, 1981. [Pg.34]

[78] Kasturi Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 641–648. ACM, 2010. [Pg.24]

[79] DW Wang and Yue-Sun Kuo. A study on two geometric location problems. *Information processing letters*, 28(6):281–286, 1988. [Pg.25]

[80] Douglas J White. The maximal-dispersion problem. *IMA Journal of Mathematics Applied in Business and Industry*, 3(2):131–140, 1991. [Pg.24]

[81] Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982. [Pg.21]

[82] Sam Chiu-wai Wong. Tight algorithms for vertex cover with hard capacities on multi-graphs and hypergraphs. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms*, pages 2626–2637, 2017. [Pg.22]

[83] G. Yao, J. Bi, Y. Li, and L. Guo. On the capacitated controller placement problem in software defined networks. *IEEE Communications Letters*, 18(8):1339–1342, 2014. [Pg.7]

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati 781039, India