Fault Tolerant Euclidean k-Centers

by

Sahar Lamey

A thesis submitted to

Faculty of Graduate and Postdoctoral Studies of

The University of Manitoba

in partial fulfillment of the requirements

of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada

October 2025

Abstract

The Euclidean k-center problem is a fundamental question in computational geometry and facility location. Given a set P of n points in $\mathbb{R}^d$, the goal is to choose a set F of k center points such that the maximum distance from any point in P to its nearest center in F is minimized. Geometrically, this corresponds to covering all points in P with k balls of minimal radius.

We study a natural generalization known as the ℓ-fault-tolerant Euclidean k-center problem, which introduces a robustness parameter $\ell \leq k$. In this variant, each point in P must be covered by at least ℓ of the k balls, or equivalently, its distance to the ℓth nearest center in F must be minimized. This captures scenarios where redundancy is required for fault tolerance or load balancing.

Our contributions include an exact $O(n \log n)$-time algorithm for solving the problem in one dimension ($\mathbb{R}$), where a linear order among points can be exploited. In two dimensions ($\mathbb{R}^2$), we prove that the problem becomes NP-hard. Nevertheless, we present an $O(nk/\ell)$-time algorithm that computes a 2-approximation, offering an efficient and practical solution with provable guarantees.

Acknowledgments

I would like to express my deepest gratitude to my supervisors, Professor Stephanie Durocher and Professor Ben Li, for their invaluable guidance, encouragement, and patience throughout the course of this research. Their insight, constructive feedback, and continuous support have been essential in shaping this thesis.

My heartfelt thanks go to Maziyar Mirzaei for his steady support and belief in me, which made this journey more meaningful. I am also deeply grateful to my parents, who, though far away for over 8 years, have supported me with love and encouragement.

Finally, I thank my colleagues and friends at the University of Manitoba for their discussions, and for making this experience enjoyable.

Contribution of Authors

A shortened version of this collaborative work conducted with my supervisors was published in the proceedings of the Canadian Conference on Computational Geometry (CCCG). The thesis itself is a single-authored work.

Contents

List of Figures

Chapter 1

Introduction

Facility location problems are fundamental in combinatorial optimization and computational geometry due to their wide applicability and rich theoretical structure. At their core, these problems involve determining optimal locations for a set (or multiset) of facilities to serve a given set of clients embedded in a metric space, with the objective of minimizing some cost function based on the distances between clients and facilities. These problems arise naturally in many practical domains, including supply chain logistics, public safety and emergency response systems, the deployment of cloud and edge computing resources, and urban planning scenarios such as the placement of schools, hospitals, or electric vehicle (EV) charging stations (e.g., [18, 21, 35]). Their relevance has only grown with the increasing complexity of modern infrastructure and the demand for responsive, equitable, and resilient service delivery systems.

One of the most studied models in this class is the *k-center problem*. The objective here is to choose k facility locations such that the maximum distance from any client to its nearest facility is minimized. This max-min objective captures a fairness criterion, ensuring that the worst-served client is still within an acceptable distance of a service point. Such guarantees are especially important in settings where equitable access is critical. For example, consider placing 10 electric vehicle

(EV) chargers to serve 1000 homes; the goal would be to ensure that every home is close to at least one charger.

However, real-world systems often experience failures or overloads. EV chargers might be busy, cloud servers can go offline, or emergency stations might be temporarily unavailable. This motivates the need for fault-tolerant facility placement models. In the *ℓ-fault-tolerant k-center problem*, each client must be within a minimal distance of not just its nearest facility, but of its ℓth nearest facility. This ensures that service can continue even in the presence of up to $\ell - 1$ facility failures.

The discrete version of the *ℓ-fault-tolerant k-center* problem was first studied by Krumke [27], who examined the case for arbitrary $1 \leq \ell \leq k$ on graphs whose edge weights define a metric space. They showed that the problem is NP-hard and that no polynomial-time $(2 - \epsilon)$-approximation algorithm is possible for any $\epsilon > 0$, unless P = NP. Krumke also provided a 4-approximation algorithm for the problem. This was subsequently improved by Chaudhuri et al. [3], who introduced a 2-approximation algorithm for general metric graphs. The version of the problem considered in both of these papers considers points which are not centers when computing the point that has the furthest distance to its ℓth closest center; i.e., the cost function evaluates the maximum distance between any input point p not selected as a center and the ℓth-nearest center to p.

Later, Khuller et al. [24] studied two natural variants of this problem. In the first, every point – including those that serve as centers – must be within a specified distance to at least ℓ centers. In the second, only non-center points are required to satisfy this coverage condition. For both variants, they presented polynomial-time approximation algorithms with constant-factor guarantees. Specifically, for the first version, they provided a 3-approximation algorithm for general ℓ and a tighter 2-approximation when $\ell < 4$. Their second version is the same as that of Krumke [27]. For this version, they provided a 2-approximation algorithm

matching the result of Chaudhuri et al. [3].

Additionally, [24] extended their framework to the fault-tolerant k-supplier problem, where centers must be selected from a specified subset of supplier nodes, and each demand node must be covered by at least ℓ centers. For this generalization, they provided a 3-approximation algorithm, which is also tight assuming $\mathrm{P} \neq \mathrm{NP}$.

Recently, Kumar and Raichel [28] studied this model and proposed simple yet powerful techniques to extend any approximate solution to the classical m-center problem into an approximate fault-tolerant k-center solution, where $m = \lfloor k/\ell \rfloor$; their key idea is to reinforce each center in the smaller non-fault-tolerant solution by adding its ℓ nearest neighbors, producing a $(1 + 2c)$-approximation, where c is the approximation factor of the original algorithm. Their second main result is that using the algorithm of Gonzalez [15] for the initial m-center solution, they give a tighter approximation ratio guarantee. Specifically, 3-approximation when $\ell | k$, and a 4-approximation otherwise, for the discrete fault-tolerant k-center problem.

Drezner [8] briefly explored the 2-fault-tolerant Euclidean 2-center problem, motivated by applications involving unreliable facilities. To the best of our knowledge, no prior work has provided general approximation algorithms for the ℓ-fault-tolerant Euclidean k-center problem for arbitrary k and ℓ. The present work aims to bridge this gap by developing approximation algorithms and geometric insights for fault-tolerant clustering in Euclidean space.

In this thesis, we focus specifically on the *ℓ-fault-tolerant Euclidean k-center problem*, where clients and facilities lie in a Euclidean space. The continuous nature of Euclidean space introduces geometric structure that can be exploited to develop efficient approximation algorithms, but it also introduces unique hardness challenges. The goal is to select k facility locations in $\mathbb{R}^d$ such that the maximum Euclidean distance from any client to its ℓth nearest facility is minimized.

We aim to show the NP-hardness and explore approximation algorithms for

this problem, analyze their performance guarantees, and understand their geometric properties.

The remainder of this thesis is organized as follows:

- **Chapter 2: Background and Preliminaries**

  This section introduces the theoretical foundations required for the study of fault-tolerant clustering problems. We define key concepts from facility location, clustering, and approximation algorithms, and formally present the ℓ-fault-tolerant Euclidean k-center problem along with relevant notation.

- **Chapter 3: Related Work**

  We survey the most relevant literature in this area, including classical results for the k-center problem, fault-tolerant extensions, and both graph-based and geometric versions. Key contributions such as those of Khuller et al., Krumke, and Kumar and Raichel are discussed in detail.

- **Chapter 4: Geometric Properties**

  This section explores the geometric structure of solutions in the Euclidean setting. And examines some simple and intuitive approaches to the problem.

- **Chapter 5: k-Center Problems on 1D**

  We first present a polynomial-time algorithm for solving the Euclidean k-center problem on the real line. We then extend this result to the fault-tolerant version, showing that even with redundancy requirements, the problem remains tractable in one dimension.

- **Chapter 6: NP-Hardness of *fault-tolerant Euclidean k-center* on 2D**

  In this section, we prove that the *fault-tolerant Euclidean k-center* problem is NP-hard in two dimensions. The reduction is constructed using a series of geometric gadgets, including variable and clause structures, and builds upon the Planar 3-SAT problem via a non-crossing rectilinear embedding.

- **Chapter 7: 2D Approximation Algorithm**

  We describe and analyze approximation algorithms for the Euclidean k-center and its fault-tolerant variant. Starting from Gonzalez's 2-approximation for the classical problem, we demonstrate how to adapt it for Euclidean fault tolerance, achieving a 2-approximation. We also evaluate the algorithm's time complexity.

- **Chapter 8: Conclusion and Future Work**

  We summarize our findings and contributions, particularly the approximation guarantee derived from reducing to a $\lfloor k/\ell \rfloor$-center instance. We also discuss the inherent difficulty of improving the approximation factor and suggest possible directions for future research, such as parameterized algorithms, dynamic updates, or extending to higher fault-tolerance models.

Chapter 2

Background and Preliminaries

2.1 Facility Location Problems

Facility location problems have been widely studied in the Operations Research and Computer Science communities (e.g., see [36] for a survey [38]). These problems have a wide range of applications in real-world scenarios, such as the placement of warehouses, emergency services, communication towers, or data centers. The main challenge in these types of problems is to make decisions about where to locate resources so that the service delivery, accessibility, and operational efficiency is optimized. In other words, the goal is to determine optimal positions for facilities to serve a given set of clients while minimizing some cost function. The typical form of this problem can be presented as follows:

- **Instance:** A set of client positions is given.
- **Question:** Find a set of positions for facilities that provide service to these clients, which minimizes a given objective function.

There are different formulations of facility location problems, each of which corresponds to specific cost functions and assumptions. Some of the most-studied

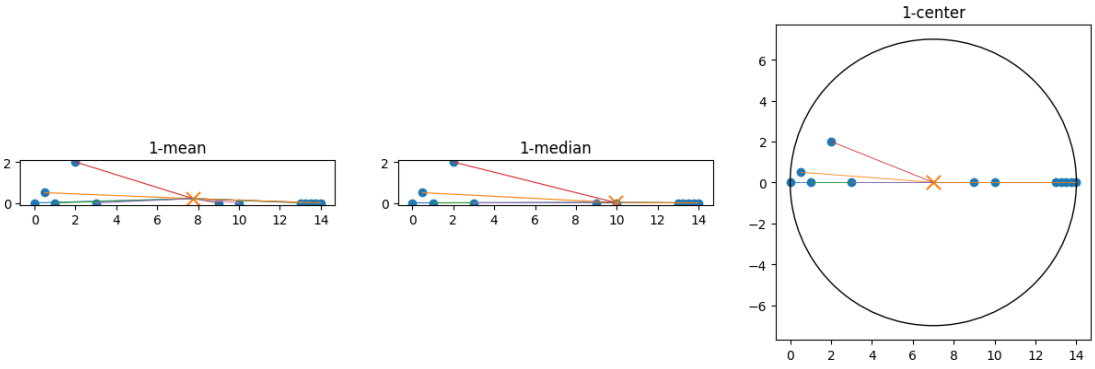

Figure 2.1: Illustration that the Euclidean 1-mean, 1-median, and 1-center problems yield distinct optimal centers on the same dataset.

variants include the k-median, k-means, and k-center problems. Here is a brief explanation on each of them to see how they are different from each other:

- ***k*-median Problem:** The goal is to locate k facilities to minimize the sum of the distance of all clients from their nearest facilities. It is commonly used in scenarios where overall service quality matters, like logistic optimization (see, e.g. [29]).
- ***k*-means Problem:** Similar to the k-median problem but minimizes the sum of squared distances of all clients from their nearest facilities. This is widely used in machine learning, data mining and pattern recognition (see, e.g. [1]).
- ***k*-center Problem:** Focuses on minimizing the maximum distance between any client and its assigned facility. This model is important in applications where the worse-case service time or distance really matters, such as emergency response systems.

The figure 2.1 shows this difference visually.

In this thesis, we mainly focus on the ***k*-center problem**. Especially, when we apply **fault-tolerant** constraints and when the clients and facilities are located in **Euclidean** space. Example applications of the k-center problem in real-world

scenarios include urban plannings like emergency responses, libraries, and the transportation industry. Another practical example would be in military logistics, such as locating missile defense systems to protect population centers. This problem is discussed in more detail in the following sections.

2.2 k-Center Problem

The classical k-center problem is one of the centeral problems in facility location, optimization, and clustering. It can be defined, as follows:

Definition 2.2.1 (k-center Problem)**.** *Let $P = \{p_1, p_2, \ldots, p_n\} \subseteq \mathcal{M}$ be a set of n client points in a metric space $(\mathcal{M}, dist)$, where $dist : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ is a distance function. The distance function is non-negative and satisfies triangle inequality. Given an integer $k \geq 1$, find a set or multiset $F = \{f_1, f_2, \ldots, f_k\} \subseteq \mathcal{M}$ of k facilities or centers such that the covering radius of P gets minimized:*

$$\min_{\substack{F \subseteq \mathcal{M} \\ |F|=k}} \left(\max_{p \in P} \min_{f \in F} \operatorname{dist}(p, f) \right). \tag{2.2.1}$$

Variants of the k-Center Problem

There are two common variants of the problem, one is the discrete k-center in which the facilities are supposed to be a subset of the client set or multiset (i.e. $F \subseteq P \subseteq \mathcal{M}$). The *discrete k-center* problem examines the corresponding problem in the graph setting, where the metric space is limited to the set of vertices of the graph. And the second variant is the continuous version or the *Euclidean k-center* in which the centers can be placed anywhere in $\mathbb{R}^d$ ($F \subseteq \mathcal{M} = \mathbb{R}^d$). Since the discrete version restricts the solution space, the respective costs of solutions to these variants can differ. The corresponding radii of covering balls for these two versions of the k-center problem can differ by a factor of two; Figure 2.2 illustrates this difference.

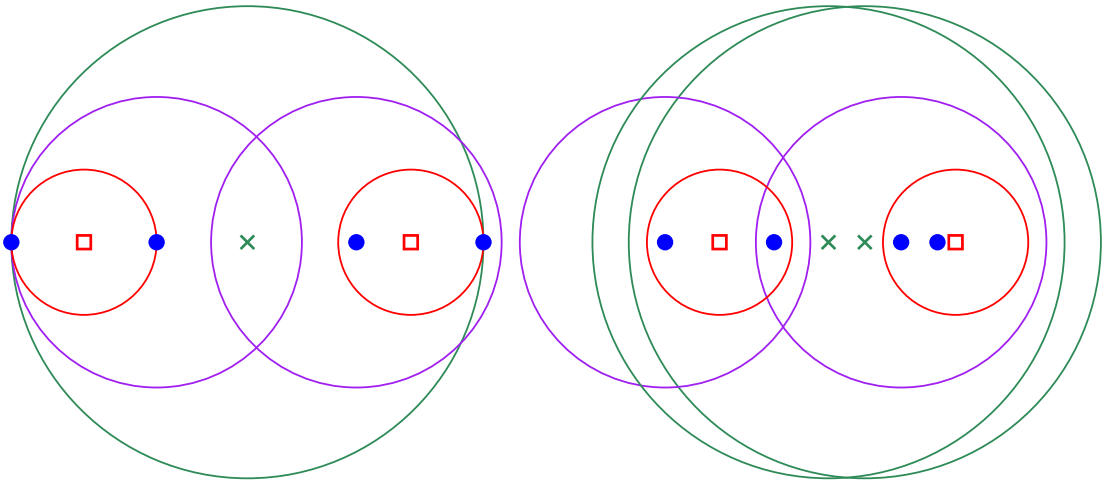

Figure 2.2: A set P of eight points (blue), a Euclidean 4-center of P (red $\square$), a discrete 4-center of P (purple), and a 2-fault-tolerant Euclidean 4-center of P (green $\times$) with two facilities collocated on the left $\times$.

2.3 Fault Tolerance in *k*-Center

In the classical k-center problem the general assumption is that all k facilities should operate consistently with no interruption, however this is not always the case in real world; facilities might be overloaded or out of service due to disruptions, natural disasters, constructions and terrorist attacks. So we need more robustness on the assigned facilities. To introduce robustness to the located facilities, the fault-tolerant version of the problem can be defined and formulated [27] which requires each client to be covered by multiple facilities. Therefore, the *ℓ-fault-tolerant k-center problem* assures that for each client point $p \in P$ there exist at least ℓ centers within some determined radius of p. The aim of this constrained version of k-center problem is to ensure if up to $\ell - 1$ centers fail to provide service, each client will be served by the ℓth facility within the minimized radius. In this thesis, we specifically concentrate on the Euclidean version of the fault-tolerant k-center problem. This problem is defined as follows [27]:

Definition 2.3.1 (ℓ-Fault-Tolerant Euclidean k-Center Problem)**.** *Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points in the Euclidean space $(\mathbb{R}^d, dist)$, and integers k, ℓ such that $1 \leq \ell \leq k$, find a multiset of k centers $F = \{f_1, f_2, \ldots, f_k\} \subseteq \mathbb{R}^d$ that*

minimizes the maximum distance from any point $p \in P$ to its ℓth nearest center in F:

$$\min_{\substack{F \subseteq \mathbb{R}^d \\ |F|=k}} \text{cost}_\ell(P, F), \tag{2.3.2}$$

where

$$\text{cost}_\ell(P, F) = \max_{p \in P} \text{dist}_\ell(p, F), \tag{2.3.3}$$

and $\text{dist}_\ell(p, F)$ *denotes the distance from p to its ℓth nearest center in F.*

$\text{dist}_\ell(p, F)$ can be formally defined as follows:

Definition 2.3.2. *Let F be a multiset of points in the Euclidean space $(\mathbb{R}^d, dist)$, and let $p \in \mathbb{R}^d$. For a positive integer $\ell \leq |F|$, the ℓth nearest center to p in the multiset F can be defined as follows:*

$$dist_\ell(p, F) = \textit{The } \ell\textit{th smallest value} \in \{dist(p, f) : f \in F\}.$$

This model is applicable in scenarios such as cloud server placement, where a client should have access to multiple backup servers, or EV charging networks, where congestion or failure of the nearest charger makes it difficult to get access to the alternative option. Another example is missile defense systems placement, where densely populated geographic coordinates of a city must be protected by a back up missile defense, in the case of failure of first few of them.

The Case $\ell = 1$: k-Center Problem

The ℓ-fault-tolerant Euclidean k-center is a generalization of the Euclidean k-center problem, each point should be covered by at least ℓ centers. When $\ell = 1$, this means we require each point to be covered by its *nearest* center. Thus, if $P \subseteq \mathbb{R}^d$ is a set of n points and $F \subseteq \mathbb{R}^d$ is the set of k selected centers, then

from (2.3.2) and (2.3.3) we can get the objective function of the Euclidean k-center problem as follows:

$$\min_{\substack{F\subseteq\mathbb{R}^d \\ |F|=k}} \text{cost}_1(P,F),$$

where

$$\text{cost}_1(P,F) = \max_{p\in P} \text{dist}_1(p,F).$$

Hence, minimizing $\text{cost}_1(P,F)$ over all $|F| = k$ is exactly the objective of the Euclidean k-center problem. And $\text{dist}_1(p,F)$ denotes the distance from p to its *nearest* center in F,

$$\text{dist}_{\ell=1}(p,F) = \min_{f\in F} \|p-f\|_2$$

Therefore, the classical k-center problem can be a special case of the ℓ-fault-tolerant k-center problem when $\ell = 1$. As ℓ increases, the problem requires additional coverage for every client.

2.4 Approximate Solutions

It is known that finding an exact solution for the Euclidean k-center problem is NP-hard [33]. With that being said, the larger the data set gets, the more intractable this optimization problem becomes. A commonly applied method would be approximation algorithms.

Definition 2.4.1 (α-approximate Solution)**.** *Let $\mathcal{P}$ be an NP-Hard optimization problem. Given an instance I of the problem $\mathcal{P}$ and an algorithm $\mathcal{A}$ that computes a feasible solution for instance I, $OPT(I)$ denotes the quality associated with an optimal solution for instance I. And $ALG_{\mathcal{A}}(I)$ denotes the quality of the solution computed by algorithm $\mathcal{A}$ (The quality of a solution is the objective value of the solution). Let $\alpha \geq 1$ be a real number.*

- *When $\mathcal{P}$ is a maximization problem, we say that a polynomial-time algorithm*

$\mathcal{A}$ is a $\frac{1}{\alpha}$-approximation algorithm for problem $\mathcal{P}$ if for all instances I of the problem $\mathcal{P}$,

$$\frac{1}{\alpha} \cdot OPT(I) \leq ALG_{\mathcal{A}}(I),$$

- *When $\mathcal{P}$ is a minimization problem, we say that a polynomial-time algorithm $\mathcal{A}$ is a α-approximation algorithm for problem $\mathcal{P}$ if for all instances I of the problem $\mathcal{P}$,*

$$ALG_{\mathcal{A}}(I) \leq \alpha \cdot OPT(I).$$

The constant α is called the approximation factor *of the algorithm.*

In the context of the ℓ-fault-tolerant Euclidean k-center problem, let F^* be an optimal solution to the objective function:

$$F^* = \operatorname{argmin}_{\substack{F \subseteq \mathbb{R}^d \\ |F|=k}} \operatorname{cost}_\ell(P, F). \tag{2.4.4}$$

Recall that $\operatorname{cost}_\ell(P, F)$ denotes the maximum distance from any point $p \in P$ to its ℓth nearest center in F.

A multiset F' of k centers is said to be an *α-approximate solution* to the ℓ-fault-tolerant k-center problem if:

$$\operatorname{cost}_\ell(P, F') \leq \alpha \cdot \operatorname{cost}_\ell(P, F^*) = \alpha \cdot \mathrm{OPT}.$$

Interpreted geometrically, covering each point in P by ℓ balls centred on points in F' requires a radius at most α times larger than the radius of balls centred on points in an optimal solution F^*.

In this chapter, we presented the necessary background to support the results and algorithms in the rest of the thesis. We also introduced the key definitions and concepts along with practical applications in real life situations.

Chapter 3

Related Work

3.1 Classical *k*-Center Problem

The k-center problem is among the fundamental optimization problems of facility location, graph algorithms, and clustering. At its core, recall that the problem asks: Given a set P of points in a metric space and a positive integer k, how can we locate k centers to minimize the maximum distance from any point in P to its nearest center?

This natural and interesting goal has led to many years of research. Over time, many new versions of the problem were created to handle real needs, such as fault tolerance, capacity limits, and geometric restrictions (see, e.g., [25, 27]).

The classical k-center problem is known to be NP-hard in general metric spaces [14, 15], meaning that no polynomial-time algorithm can guarantee to solve all instances of the problem optimally unless P = NP. This hardness remains even when the input points lie in the Euclidean plane or on graphs, and the difficulty increases with higher-dimensional and larger values of input [23, 33].

Hochbaum and Shmoys (1985) [16] established that it is NP-hard to approximate the k-center problem within any factor better than 2 in an arbitrary metric

space, setting a lower bound on the approximation factor that can be guaranteed by any algorithm. Despite this, efficient algorithms exist that match this bound, providing 2-approximations in polynomial time (see e.g. [17]).

Feder and Greene [10] proved that for dimensions $d \geq 2$, the Euclidean k-center problem cannot be approximated within a factor better than $\frac{1+\sqrt{7}}{2} \approx 1.822$, unless P = NP. This result shows that even with the geometric setting of Euclidean space, the problem is still computationally hard once we go beyond a certain approximation threshold. In addition to this hardness result, they presented a 2-approximation algorithm that runs in $O(n \log k)$ time and also showed this running time is optimal in the algebraic decision tree model.

3.2 2-Approximation Algorithms

The best-known 2-approximation algorithm for the k-center problem is the greedy algorithm introduced by Gonzalez [15]. The method proceeds by first choosing an arbitrary input point as the initial center. Then, it repeatedly selects the point that is farthest from any center among the set of centers currently chosen, adding that point to the center set of centers, until k centers have been selected. This greedy strategy guarantees that no client is farther than twice the optimal radius from its nearest selected center.

Another important 2-approximation method was given by Hochbaum and Shmoys [17]. Their idea is to perform a binary search on the radius r and try to find the smallest r so that the input can be covered with k balls of radius r. For each iteration, the algorithm checks feasibility by reducing the problem to a set cover instance and then applying an appropriate cover algorithm. The smallest feasible r from the search gives a covering with guaranteed approximation quality.

Both algorithms achieve the best possible approximation factor under standard complexity assumptions. Their designs are simple yet powerful, and they form

the basis for numerous extensions of the k-center problem, including capacitated, weighted, outlier-tolerant, and fault-tolerant versions (see, e.g., [25, 27]).

3.3 Exact Algorithms for Special Cases

Although the general k-center problem is hard to solve exactly, there exist polynomial-time exact algorithms for restricted cases and for constant values of k. In fact, several variants become more manageable under additional constraints, and these restricted versions have been studied in detail. In the next section, we go over some of these restricted cases.

3.3.1 One-Dimensional Case

When the input points lie on the real line (i.e., the 1D case), the k-center problem can be solved exactly in $\mathcal{O}(n \log n)$ time. The 1D k-center problem can be solved in $\mathcal{O}(n \log n)$ time by combining the techniques of Megiddo and Tamir [34] with Cole's parametric search [7]. Although the method in [34] is efficient and straightforward, Cole's parametric search [7] is complicated and involves large constant factors in its running time. Chen and Wang [5] present another $\mathcal{O}(n \log n)$-time algorithm for 1D k-center, which is easier and avoids parametric search. Moreover, if all points in P are given in sorted order on the real line and their weights are also sorted, their algorithm can solve the 1D k-Center in

$$\mathcal{O}\left(n + k^2 \log^2 \frac{n}{k} \log n \log\log n\right)$$

time, which is particularly efficient for small values of k. For example, if $k = \mathcal{O}(n^{1/2-\epsilon})$ for any $\epsilon > 0$, then their algorithm runs in linear time $\mathcal{O}(n)$. In fact, Chen and Wang [5] also established an $\Omega(n \log n)$-time lower bound for the problem when the input set P is unsorted and k is arbitrary.

3.3.2 Tree Metrics

In graph settings where the metric space is limited to the vertices of a tree graph, assuming that every vertex has a positive weight associated with it, and distances are defined over a tree graph as the length of the unique path in between two nodes, the problem of k-center becomes tractable. Kariv and Hakimi [22] provided a dynamic programming-based algorithm that solves the k-center problem on trees in polynomial time. This result is particularly significant for hierarchical network models, such as utility grids and telecommunication infrastructures.

Megiddo and Tamir [34] presented an $\mathcal{O}(n \log^2 n \log \log n)$-time algorithm for the weighted k-center problem on a tree with n nodes, and this running time can be reduced to $\mathcal{O}(n \log^2 n)$ by applying Cole's parametric search [7]. Later, Frederickson [13] gave a linear time algorithm for the unweighted k-center problem on a tree. Jeger and Kariv [20] proposed an $\mathcal{O}(kn \log n)$-time algorithm for the weighted k-center problem on a tree.

In addition, the discrete weighted k-center problem on a tree can be solved in $\mathcal{O}(n \log^2 n)$ time [34], and the discrete unweighted k-center problem on a tree can be solved in $\mathcal{O}(n)$ time [13].

3.3.3 Small Fixed k in Euclidean Space

For small values of k, such as $k = 1$ or $k = 2$, efficient exact algorithms exist in the Euclidean plane. In the special case where $k = 1$, the problem can be solved in $O(n)$ time [31, 32], and this result not only holds in the plane but also in any fixed dimension. A simple $O(n)$-time randomized algorithm was also given by Welzl [40].

Drezner [8] and later Sharir and Eppstein [37] developed algorithms that solve the Euclidean 2-center problem in the plane in near-linear time. Sharir and Eppstein [37] presented a novel algorithm for solving the 2-center problem. The al-

gorithm runs in $\mathcal{O}(n \log^9 n)$ time, improving the previous $\mathcal{O}(n^2 \log n)$ time algorithm [19], and providing the first subquadratic-time algorithm for this problem. These methods rely on geometric techniques such as farthest-point Voronoi diagrams and arrangements of disks to compute the minimum covering radius for two centers. Eppstein [9] and Chan [2] further improved this result, obtaining an $\mathcal{O}(n \log^2 n)$-time randomized algorithm and an $\mathcal{O}(n \log^2 n \log \log n)$-time deterministic algorithm, respectively. First, Wang [39], gave an $\mathcal{O}(n \log^2 n)$-time deterministic algorithm. Recently, Wang[6], presented an $\mathcal{O}(n \log n)$-time deterministic algorithm for the Euclidean 2-center problem in the plane; this is asymptotically optimal, since there is a worst-case lower bound of $\Omega(n \log n)$ time.

The provided solutions for special cases of the problem are not broadly applicable to all instances, but they can give valuable insights. In fact, they form the basis of tackling the problems and developing approximation and heuristic methods in more general settings.

3.4 From k-Center to Fault-Tolerant k-Center

The classical k-center problem assumes that all chosen centers remain functional and available to serve clients. However, real-world systems often encounter failures, congestion, or planned outages. This motivates the **fault-tolerant k-center problem**, in which each client must be served not just by its nearest center, but by multiple nearest centers.

In the ℓ-fault-tolerant version of the k-center problem, each client must be within a fixed radius of at least ℓ distinct centers. This extension increases the problem's complexity, but provides greater robustness on the assigned facilities. The objective is to minimize the maximum distance from any client to its ℓ-th nearest facility:

$$\text{cost}_\ell(P, F) = \max_{p \in P} \text{dist}_\ell(p, F),$$

where F is a set or multiset of k facilities in $\mathbb{R}^d$, and $\text{dist}_\ell(p, F)$ denotes the distance from client p to its ℓ-th closest facility in F.

Most of the earlier work on the fault-tolerant k-center problem deals with the discrete version, where the centers must be selected from the input points. In the fault-tolerant setting, the idea is to add robustness by ensuring that each client is covered by multiple centers.

The discrete version of the ℓ-fault-tolerant k-center problem was first introduced by Krumke [27], who studied the case for arbitrary $1 \leq \ell \leq k$ on graphs where the edge weights define a metric space. He showed that the problem is NP-hard and proved that no polynomial-time $(2 - \epsilon)$-approximation algorithm exists for any $\epsilon > 0$, unless P = NP. As a positive result, Krumke provided a 4-approximation algorithm for the problem. This guarantee was later improved by Chaudhuri et al. [3], who developed a 2-approximation algorithm. Both of these works considered the version of the problem where only non-center vertices are evaluated when computing the maximum distance to the ℓ-th nearest center.

Khuller [24] studied two variants of the fault-tolerant k-center problem on graphs. In the first variant, every vertex is required to have at least ℓ centers within the specified radius. In the second variant, only the vertices that are not chosen as centers must be within ℓ centers. For both variants, they designed polynomial-time approximation algorithms with constant-factor approximation. Specifically, for the first variant they achieved an approximation factor of 3 for arbitrary ℓ, and improved this to 2 when $\ell < 4$. For the second variant, they gave a 2-approximation for all values of ℓ, which is essentially optimal since even the classical k-center problem cannot be approximated below a factor of 2 unless NP = P.

Kumar and Raichel [28] introduced a framework that connects the non-fault-tolerant and fault-tolerant versions of clustering problems. They proposed a technique that transforms any constant-factor approximation for the standard m-center problem (where $m = \lfloor k/\ell \rfloor$) into a fault-tolerant solution for the k-center problem. They showed that by selecting m centers using a standard algorithm like Gonzalez's [15] and extending each with its ℓ nearest neighbours, one obtains a $(1+2c)$-approximation, where c is the approximation factor for the m-center subroutine. When Gonzalez's 2-approximation is used, this yields a 3-approximation if ℓ divides k, and a 4-approximation otherwise. The approximation guarantees obtained by their method are not optimal, but the strength of the approach lies in its simplicity and ease of use.

3.4.1 Fault-tolerant Clustering Under Capacity Constraints

Another important line of research explores capacitated versions of the fault-tolerant k-center problem, where each service center can serve only a limited number of clients. In this setting, failures become more challenging because clients must be reassigned while still taking the capacity limits of surviving centers into account. The problem first studied by Chechik and Peleg [4], considering two variants of the problem.

The first, called the α-fault-tolerant capacitated k-center (α-FT-CKC) problem, allows complete reassignment after failures: if up to α centers fail, all clients may be reassigned to surviving centers as long as capacities are respected. This version provides flexibility in handling failures, but requires careful balancing of loads across centers in advance.

The second, called the α-fault-tolerant conservative capacitated k-center (α-FT-CCKC) problem, is similar to the α-FT-CKC problem, but more restrictive. Here, after up to α centers fail, only the clients originally assigned to the failed

centers are allowed to move. Clients whose centers remain active are not reassigned, even if rebalancing could reduce the overall cost. This conservative rule makes the problem significantly harder, as fewer options exist to redistribute load when failures occur.

Capacitated fault-tolerant k-center is harder than uncapacitated fault-tolerant k-center because it requires planning not only for coverage under failures but also for respecting client load limits, which restricts how clients can be reassigned after failures.

Chechik and Peleg [4] presented polynomial-time approximation algorithms for both variants of the problem. In particular, they obtained a 9-approximation for the α-FT-CKC problem and a 17-approximation for the α-FT-CCKC problem. Their algorithms for both versions are based on using the (non-fault-tolerant) algorithms proposed by Khuller and Sussmann [25] as a starting point, and introducing the necessary modifications to make them tolerant against the failure of some centers. Later, Fernandes [11] proposed a new approach that first selects and pre-opens a set of backup centers, and then solves the remaining part of the problem as a residual instance. The idea of pre-opening is to guarantee that each cluster already has α centers with enough backup capacity, so failures can be handled locally. After this step, the rest of the problem can be solved either by reducing it to the non-fault-tolerant version or by using an LP relaxation with rounding. For the special $\{0, L\}$-capacitated case, this method achieved a 6-approximation for the α-FT-CKC problem and a 7-approximation for the conservative variant, improving the earlier bounds of 9 and 17, respectively.

3.4.2 ℓ-fault-tolerant Euclidean k-Center

In contrast to the body of work on the discrete setting, the *ℓ-fault-tolerant Euclidean k-center* problem has received relatively limited attention. In this version,

the facilities are not restricted to input points but can be placed anywhere in $\mathbb{R}^d$. This flexibility introduces greater algorithmic and geometric challenges.

Chapter 4

Geometric Properties

The optimal solution to the classical k-center problem is not necessarily unique. There can be multiple distinct sets of facility locations that achieve the same minimum cost. For example, consider client points located at the four corners of a square in the plane, where each side has length a. For $k = 2$, the optimal covering radius is $\frac{a}{2}$, and this can be achieved by at least two different solutions: one where the two facilities are placed at the midpoints of two opposite vertical sides, and another where they are placed at the midpoints of the horizontal sides (see Figure 4.1). Similarly, with ℓ-fault-tolerant Euclidean k-center F^* is not unique in general, but $\text{cost}_\ell(P, F^*)$, that is, the minimum radius needed to cover each point with at least ℓ centers is equal for all F^* that minimize (2.4.4). To better understand this optimization objective it helps to interpret the ℓ-fault-tolerant k-center problem as a *ball-coverage* problem.

4.1 Interpretation of ℓ-fault-tolerant k-Center as Ball Coverage

Recall that $\text{dist}(u, v)$ denotes the Euclidean (ℓ_2) distance between the points $u, v \in \mathbb{R}^d$. Equivalently, $\text{dist}(u, v)$ is the radius of the smallest ball centred at u that

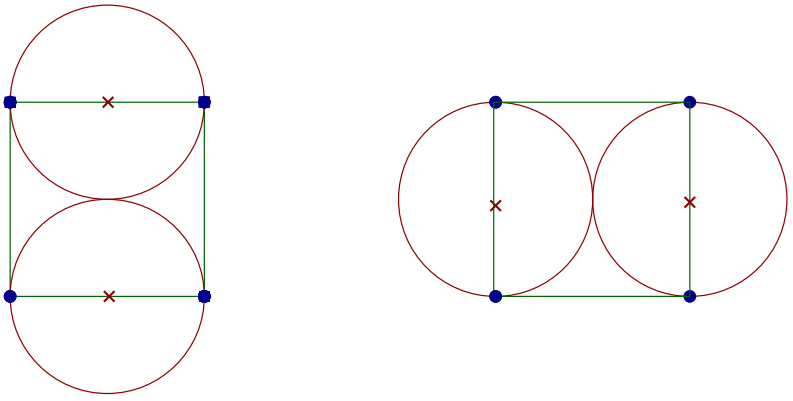

Figure 4.1: The blue points are client points located at the four corners of a square with side length a. The crosses represent two different valid placements of the two centers: one pair covering the midpoints of opposite vertical sides, and another pair covering the midpoints of opposite horizontal sides. Both solutions achieve the same optimal covering radius of $\frac{a}{2}$.

contains v. Let $S \subseteq \mathbb{R}^d$ be a multiset of k facility locations. The ℓ-fault-tolerant k-center problem determines the minimum radius r such that every point $p \in P \subseteq \mathbb{R}^d$ lies within distance r of at least ℓ facilities in S.

Formally, find:

$$\min_{S \subseteq \mathbb{R}^d, |S|=k} \max_{p \in P} \text{dist}_\ell(p, S),$$

where $\text{dist}_\ell(p, S)$ is the distance to the ℓ-th closest facility in S, and S is a multiset meaning that multiple facilities are allowed to be placed at the same location.

Equivalently, in terms of Ball-Coverage, the goal is to find the smallest r, there exists a set S of k points such that:

$$\forall p \in P, \quad |\{f \in S : \|p - f\|_2 \leq r\}| \geq \ell.$$

Repetitions are allowed in S, and this distinguishes the Euclidean version from the discrete one. As in the discrete version, multiplicities are not allowed, and the centers must be chosen from the set of given client points.

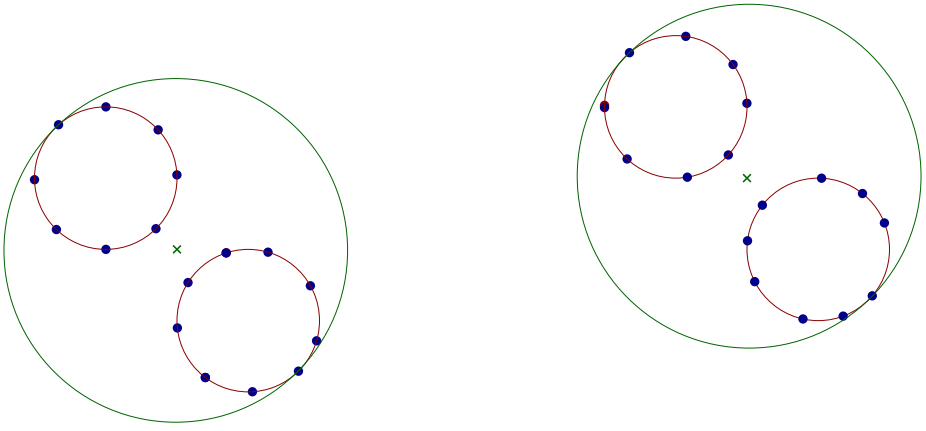

Figure 4.2: (green $\times$) is 2-fault-tolerant Euclidean 4-center of the set P of dark blue client points $\bullet$, where each facility must be duplicated. In this case, the optimal objective value for the 2-fault-tolerant Euclidean 4-center problem is the same as the optimal objective value for the Euclidean 2-center problem.

4.1.1 Building on Classical Solutions

When $\ell > 1$, an ℓ-fault-tolerant Euclidean k-center is a multiset that may require collocating multiple facilities at a common point to ensure that each client point is served by more than one facility (see Figure 2.2).

One simple and intuitive idea to build an ℓ-fault tolerant solution would be to first solve the classical $\lfloor \frac{k}{\ell} \rfloor$-center problem, without considering the fault-tolerant constraint. Then, by putting ℓ copies of each of those facility locations, we get the total of $\ell \cdot \lfloor \frac{k}{\ell} \rfloor \leq k$ facility locations, which satisfies the budget of k facilities. A similar technique has been discussed in the work of Kumar and Raichel [28], though our analysis and applications in Chapters 5 and 7 extend and adapt this approach to our setting.

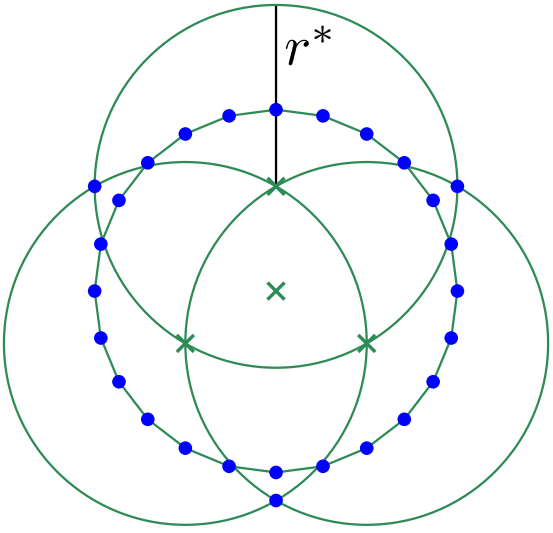

Figure 4.3: The set P of blue points has a 2-fault-tolerant Euclidean 4-center (green $\times$) of radius r^*; all Euclidean 2-centers of P have radius strictly greater than r^* [8].

4.1.2 Limitations of Classical Solutions

Although the proposed natural strategy in the previous section to solve the ℓ-fault-tolerant Euclidean k-center sounds intuitive and simple, this approach does not always give the best possible (optimal) solution, particularly, in 2-dimensional Euclidean space ($\mathbb{R}^2$). As shown by Drezner [8], there are instances where a better replacement of facilities – possibly using different, non-collocated locations – can achieve a smaller maximum distance from clients to their ℓth-nearest facilities. Illustrative configurations demonstrating this limitation are provided in Figures 4.3 and 4.4.

This insight is formalized in the following observation:

Observation 4.1.1 (Drezner 1987 [8])**.** $\exists P \subseteq \mathbb{R}^2$, $k \in \mathbb{R}^+$, $\ell \in \mathbb{R}^+$ *such that no* ℓ *copies of any Euclidean* $\lfloor k/\ell \rfloor$*-center of* P *is an* ℓ*-fault-tolerant Euclidean* k *center of* P*.*

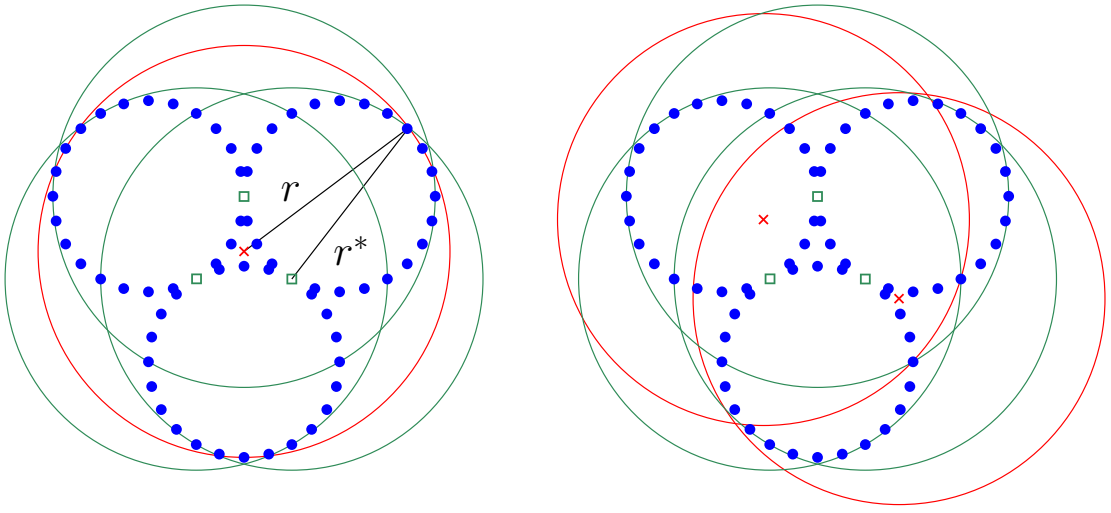

Figure 4.4: The set P of blue points has a 2-fault-tolerant Euclidean 6-center (green □) of radius r^*, but all Euclidean 3-centers of P (e.g., red ×) have radius $r = (1/\sqrt{3} + 1/2)r^* \approx 1.0774r^*$.

4.1.3 Gap Between Classical and Fault-Tolerant Solutions

Finally, we state another limitation of trying to solve the ℓ-fault-tolerant Euclidean k-center problem by just using a solution to the k-center problem.

Suppose to build an ℓ-fault-tolerant solution, we first solve the Euclidean k-center problem, which gives us exactly k facility locations, but does not consider the fault tolerance constraint. Then try to turn it into an ℓ-fault-tolerant Euclidean k-center solution by locating facilities of F on a k-center of P, which can result in $\text{cost}_\ell(P, F)$. This might appear to be a good approach, but it can actually lead to bad results in some cases. That is, $\text{cost}_\ell(P, F)$ can be arbitrarily larger than

$$r^* = \text{cost}_\ell(P, F^*),$$

for an ℓ-fault-tolerant Euclidean k-center F^* of P. That is, a Euclidean k-center cannot guarantee any approximation of an ℓ-fault-tolerant Euclidean k-center when $\ell > 1$ (see Figure 4.5).

This insight can be stated as the following observation:

Observation 4.1.2. *$\forall \alpha \in \mathbb{R}$, $\exists P$ such that $|P| = 4$ and $cost_2(P, F) > \alpha \cdot OPT$,*

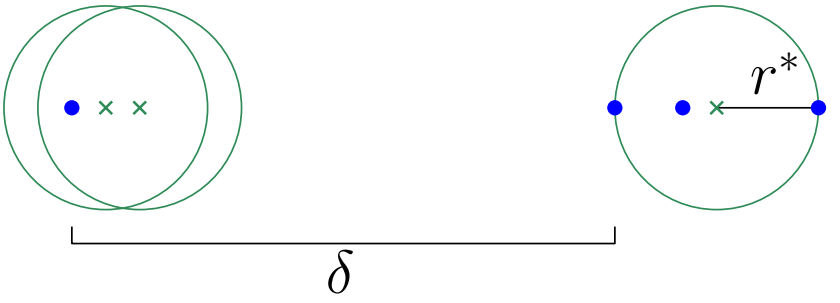

Figure 4.5: The set P of four blue points has a 2-fault-tolerant Euclidean 4-center of radius r^* (green $\times$) with two facilities collocated on the right $\times$. Setting F to a Euclidean 4-center of P ($F = P$ in this case) gives $\text{cost}_2(P, F) = \delta$, which can be arbitrarily larger than r^*.

where F is a Euclidean 4-center of P, F^* is a 2-fault-tolerant Euclidean 4-center of P, and $OPT = cost_2(P, F^*)$.

In this chapter, we illustrated through examples and observations that placing ℓ facilities at each of the $\lfloor \frac{k}{\ell} \rfloor$ facility locations obtained from classical Euclidean k-center would not always yield the optimal (smallest) ℓth radius in the fault-tolerant ball coverage model. As we show in Chapter 7, this solution guarantees a 2-approximation solution. And we also showed that locating k centers for the Euclidean ℓ-fault-tolerant k-center problem on the optimal facility locations of the classical Euclidean k-center can lead to a significantly worse cost.

Chapter 5

k-Center Problems in 1D

The problem of k-center in 1D is the easiest version of the problem, and is solvable exactly in polynomial time. In this version, all client points and facilities are on the line $\mathbb{R}$ and the distance between two points is simply the absolute difference of their values.

5.1 Polynomial-time Algorithm for Euclidean k-Center in 1D

In this section, we include a brief overview of the previous algorithms for solving the Euclidean k-center problem in one dimension, where the set of client points P lies in $\mathbb{R}$ [5, 7, 12].

Chen and Wang [5] consider the problem of finding k-centers for n weighted points on a real line. In the weighted version of the problem, each client point $p_i \in P$ is associated with a weight $w_{p_i} \geq 0$. The objective is to place k centers such that the maximum weighted distance

$$\max_{p_i \in P} w_{p_i} \cdot \text{dist}_1(p_i, F)$$

is minimized, where F denotes the set of chosen centers.

This (weighted) k-center problem was solved in $O(n \log n)$ time previously by using parametric search and other complicated approaches [7, 13], (The unweighted version of k-center in 1D is the case where all points have equal weight). Fournier and Vigneron [12] and Chen and Wang [5] present an easier $O(n \log n)$-time algorithm that avoids parametric search, and in certain special cases solves the problem in $O(n)$ time. These methods can be extended to handle the ℓ-fault-tolerant Euclidean k-center problem in one dimension, by first solving the standard k-center problem with $k' = \lfloor \frac{k}{\ell} \rfloor$ centers, and then placing ℓ copies of each center at the computed locations. Unlike in two dimensions, as we show in Theorem 5.3.2, this strategy always gives an optimal ℓ-fault tolerant k-center in one dimension.

5.2 k-Center Problem in 1D as Min-Max Partitioning

Given n distinct and increasingly sorted points $P = \{p_1, ..., p_n\} \subseteq \mathbb{R}$, the goal is to place k centers on the line such that the **maximum distance** from any $p \in P$ to its **nearest center** gets minimized (this is the k-center problem with sorted input). Equivalently, we want to partition P into continuous intervals through $k-1$ cuts, $\{i_1, i_2, ..., i_{k-1}\}$. Note that a cut is an index i_j $(1 \leq j \leq k-1)$ that separates the points of P into two consecutive groups: $\{p_{i_{j-1}}, \ldots, p_{i_j}\}$ and $\{p_{i_j+1}, \ldots, p_{i_j+1}\}$, and $i_0 = 1$ and $i_k = n$. After placing $k-1$ cuts, we obtain exactly k consecutive intervals, and we then place one center at the midpoint of each interval. This the minimizes the longest distance from a point to its assigned center. Thus, we need to partition the sorted points into k consecutive intervals, then place a center in each interval to minimize the maximum covering radius.

A related geometric problem is that of **fitting a step function** to a set of points. An optimal $\Theta(n \log n)$-time algorithm exists for the general case, and a faster $\Theta(n)$-time solution is possible when the points are already sorted by their x-coordinates. Fournierand and Vigneron in [12], Chen, Li and Wang in [5] showed a relation between k-center in 1D and the reduced instance of the following problem:

Definition 5.2.1 (Fitting a Step Function Min-ϵ Problem)**.** *Let $P' = \{p'_1, p'_2, \ldots, p'_n\}$ be a set of n points in the plane, where each $p'_i = (x_i, y_i)$. Assume the points are sorted increasingly by their x-coordinates. A* step function *g is a piecewise constant function consisting of horizontal line segments. The vertical distance between a point p'_i and the step function g is defined as $|y_i - g(x_i)|$. The* approximation error *of g with respect to P' is defined as $e(P', g) = \max_{p'_i \in P'} |y_i - g(x_i)|$. Given an integer $k > 0$, the* ***Fitting a Step Function Problem*** *asks for a step function g with at most k horizontal segments that minimizes $e(P', g)$.*

Chen, Li and Wang showed in [5] that the 1D k-center problem can be reduced to the Step Function problem. Consequently, any efficient algorithm for Step Function can be used to solve the 1D k-center problem as well. Consider the demand point set $P = \{p_1, p_2, \ldots, p_n\}$ for 1D k-center, where the points are ordered increasingly by their coordinates along the line L. For each demand point $p_i \in P$, with $1 \leq i \leq n$, they construct a new point $p'_i = (i, L(p_i))$ in the 2D Euclidean plane $\mathbb{R}^2$. Here, the x-coordinate of p'_i in $\mathbb{R}^2$ is simply the index i, and its y-coordinate is the coordinate of p_i on L. Let P' denote the set of these n points in $\mathbb{R}^2$. Lemma 1 [5], establishes the connection between the 1D k-center problem and the corresponding reduced instance of the Step Function problem.

A straightforward brute-force approach to find the optimal $\epsilon = e(P', g)$ would be to compute all $\mathcal{O}(n^2)$ such pairwise differences, sort them all in $\mathcal{O}(n^2 \log n)$-time, and then use the decision algorithm together with a binary search to find the minimum feasible ϵ. At each binary search step, run the decision algorithm. The decision test checks feasibility for a given ϵ in $\mathcal{O}(n)$. So the total for the

binary search step is:

$$\underbrace{\mathcal{O}(\log(n^2))}_{\#\text{ of guesses}} \times \underbrace{\mathcal{O}(n)}_{\text{each decision check}} = \mathcal{O}(n \log n).$$

Therefore, the total exhaustive search cost is dominated by the sorting step, giving $\mathcal{O}(n^2 \log n)$ total time.

To improve this, an efficient $\mathcal{O}(n \log n)$-time algorithm was introduced by Fournier in [12], based on ideas from [13]. This algorithm avoids generating and sorting all the $\mathcal{O}(n^2)$ values. Instead, it uses a special structure called a *sorted matrix* $M(P) = (m_{ij})$. This matrix is not stored fully in memory. Each entry m_{ij} can be calculated quickly in $\mathcal{O}(1)$ time when needed. The matrix is built in a way that each row is sorted from left to right, and each column is sorted from top to bottom. Because of this structure, the total running time is reduced to $\mathcal{O}(n \log n)$ time, which is much better than the $\mathcal{O}(n^2 \log n)$-time method. More details on this method and the matrix structure can be found in [12, 13]. This is a significant improvement over the $O(n^2 \log n)$-time approach and avoids the complexities of parametric search.

In the following section, the algorithm is extended to handle the ℓ-fault-tolerant Euclidean k-center problem in one dimension.

5.3 Polynomial-time Algorithm for Fault Tolerant k-Center in 1D

In this section, we present an algorithm for solving the ℓ-fault-tolerant Euclidean k-center problem in one dimension, where the set of client points P lies in $\mathbb{R}$. The algorithm runs in $O(n \log n)$ time and returns an optimal solution.

The key idea is simple and builds on the structure of the problem. First, we compute a solution to the classical $\lfloor k/\ell \rfloor$-center problem on the set P. This gives

us a small number of center locations that would be used if fault tolerance were not required. To achieve fault tolerance, we place ℓ copies (or facilities) at each of these center locations. This results in a total of k facilities, and ensures that each client is within the required radius of at least ℓ centers.

Because the input points lie in one dimension, the structure of the optimal solution can be fully exploited. We show that this approach gives an exact (optimal) solution in $\mathbb{R}$, while keeping the algorithm efficient and easy to implement.

Lemma 5.3.1. *Given a set P of n points in $\mathbb{R}$, and integers k, ℓ such that $1 \leq \ell \leq k \leq n$, if Z is an ℓ-fault-tolerant Euclidean k-center of P with $d_Z = cost_\ell(P, Z)$, then there exists a multiset Y that is a Euclidean k'-center of P with $cost_1(P, Y) \leq d_Z$, where $k' = \lfloor k/\ell \rfloor$.*

Proof. Let $Z = \{\{z_1, z_2, ..., z_k\}\}$ be a multiset, where $z_1 \leq z_2 \leq \cdots \leq z_k$. Define $Y = \{\{y_1, y_2, ..., y_{k'}\}\}$ where $y_i = z_{i\ell}$, for $i = 1$ to k'. Let $p \in P$. We will now show that $\text{dist}_1(p, Y) \leq d_Z$. Since Z is an ℓ-fault-tolerant k-center of P with cost d_Z, there exist (at least) ℓ consecutive elements of Z that are within distance of d_Z from p. That is, there exists i such that $\text{dist}(p, z_j) \leq d_Z$ for $j = i, i+1, ..., i+\ell$. By definition of Y, one of these z_j is in Y. Therefore, $\text{dist}_1(p, Y) \leq d_Z$. □

Theorem 5.3.2. *Given a set P of n points in $\mathbb{R}$, and integers k, ℓ such that $1 \leq \ell \leq k \leq n$, an optimal ℓ-fault-tolerant Euclidean k-center can be computed in $O(n \log n)$ time.*

Proof. Let $k' = \lfloor k/\ell \rfloor$. The algorithm computes and returns ℓ copies of S, a Euclidean k'-center of P.

Let ALG denote the cost of the solution returned by the algorithm and let OPT denote the cost of an optimal solution $\mathcal{O}$. Note that the cost of the Euclidean k'-center solution S is ALG. By Lemma 5.3.1, we can obtain from $\mathcal{O}$ a k'-center solution C with cost at most OPT. Since S is an optimal k'-center with cost

ALG, we see that $OPT \geq ALG$. Since $ALG \geq OPT$ by definition, we conclude that $ALG = OPT$ and the solution computed by the algorithm is optimal.

The running time of the algorithm corresponds to the time required to compute a k'-center of P. When $P \subseteq \mathbb{R}$, a Euclidean κ-center of P can be computed in $O(n \log n)$ time for any κ, where $n = |P|$ [5]. Consequently, our algorithm computes an ℓ-fault-tolerant Euclidean k-center of P in $O(n \log n)$ time. □

In this chapter, we studied efficient algorithms for solving the Euclidean k-center problem and its ℓ-fault-tolerant version in one dimension. We mentioned that the 1D k-center problem can be reduced to the fitting step function problem, which allows us to use algorithms that avoid complex methods like parametric search. By using a sorted matrix and a greedy feasibility check, we can solve the problem in $O(n \log n)$ time. For the fault-tolerant case, we explained that solving a smaller k'-center problem and placing ℓ copies of each center gives an exact solution. These results show that the 1D structure makes the k-center and fault-tolerant k-center problems easier to solve in an efficient way.

Chapter 6

NP-hardness of Euclidean k-center in 2D

In this chapter, we establish it is NP-hard to compute an exact ℓ-fault tolerant k-center in general. We show its NP-hardness by constructing a reduction from the Planar 3-SAT problem, which itself is known to be NP-complete [30]. We show how any instance of Planar 3-SAT can be transformed into an instance of ℓ-fault tolerant Euclidean k-center in polynomial time. This implies there exists no efficient algorithm which solves the problem exactly unless P = NP.

6.1 Planar 3-SAT

An instance $\Phi = (U, E)$ of *3-SAT* consists of a set of variables

$$U = \{u_1, \ldots, u_n\},$$

and a set of clauses

$$E = \{E_1, \ldots, E_s\}.$$

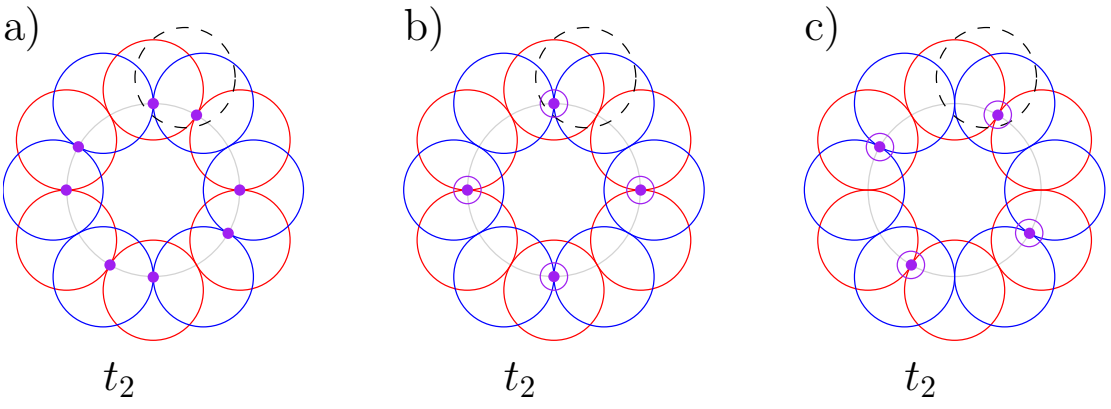

Figure 6.1: **Variable Gadget (true)**: These three configurations of eight purple points stab each red or blue disc with two points, as well as the dashed disc with two points. Pairs of purple points are collocated in the second and third configurations. The dashed black disc is part of a clause gadget; it contains two purple points.

Each clause is a disjunction of three literals, where each literal is either a variable in U or its negation. For example, a clause may be written as

$$E_1 = (u_1 \vee \neg u_2 \vee u_3),$$

which is satisfied if at least one of the literals u_1, $\neg u_2$, or u_3 evaluates to true. The objective of 3-SAT is to determine whether there exists an assignment of truth values to the variables in U such that all clauses in E are satisfied (evaluate to true).

Every instance Φ of 3-SAT can be represented as a bipartite graph G_Φ, where the vertex set is $U \cup E$, and there is an edge (u_i, E_j) if and only if the variable u_i appears in the clause E_j. In this representation, one part of the bi-partition consists of the variables, and the other part consists of the clauses.

Planar 3-SAT is a restricted version of 3-SAT in which the corresponding bipartite graph G_Φ is planar, meaning it can be embedded on the plane without any edge crossings. Planar 3-SAT remains NP-complete [30]. Furthermore, for every instance Φ of Planar 3-SAT, there exists a non-crossing rectilinear embedding of G_Φ in $\mathbb{R}^2$ [26].

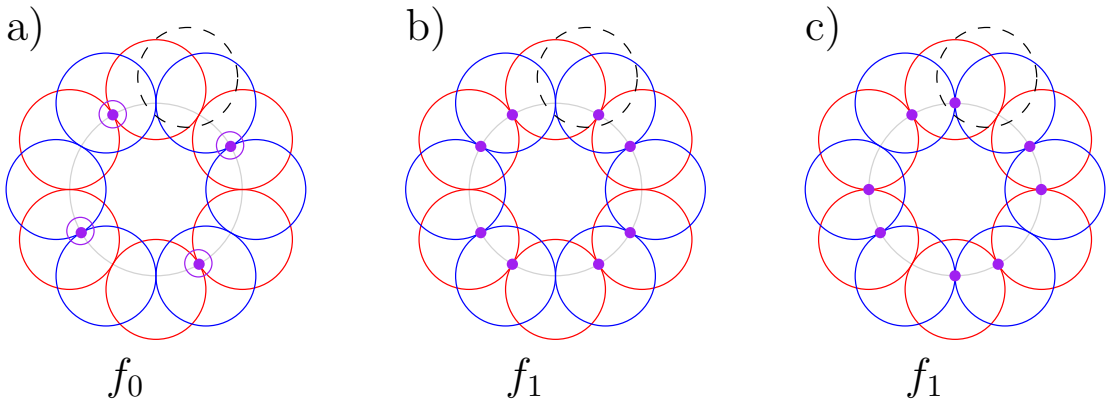

Figure 6.2: **Variable Gadget (false)**: These three configurations of eight purple points stab each red or blue disc with two points, but stab the dashed disc with only one (f_1) or zero (f_0) points. Pairs of purple points are collocated in the first configuration. The dashed black disc is part of a clause gadget; it contains zero or one purple points.

6.2 Reduction from Disc 2-Stabbing

We establish the NP-hardness of the fault-tolerant Euclidean k-center problem by showing that the *Disc 2-Stabbing* problem is NP-hard. In this problem, the input is a set P' of unit discs in $\mathbb{R}^2$ together with an integer $k \geq 1$. The goal is to decide whether there exists a multiset F of k points such that every disc in P' contains at least two points from F. If such a set F exists, then we say that F *2-stabs* P'. Notice that this condition is equivalent to requiring F to be a 2-fault-tolerant Euclidean k-center of the set of disc centers in P', with covering radius at most 1. That is, Disc 2-Stabbing reduces directly to ℓ-fault-tolerant Euclidean k-center decision problem.

We now present a reduction from the Planar 3-SAT problem to the Disc 2-Stabbing problem. Let $\Phi = (U, E)$ be an arbitrary instance of Planar 3-SAT, where U is the set of variables and E is the set of clauses. We begin by constructing a rectilinear planar embedding of the corresponding graph G_Φ, which allows us to build the gadgets needed for the reduction. Following sections explains structure of gadgets and the reduction in detail.

6.3 Variable Gadget

For each variable $u_i \in U$, we construct a cycle of $r_i = 3m_i$ overlapping discs that follows the edges adjacent to u_i in the drawing of G_Φ, where m_i is even (see Figures 6.1 and 6.2); let $C^i = \{C^i_0, C^i_1, ..., C^i_{r_i-1}\}$ denote this set of discs, where indices are taken modulo r_i. E.g., C^i_j denotes $C^i_{j \bmod r_i}$. Position the discs in C^i such that $C_i \cap C_j \neq \varnothing$ if and only if $(i-j) \bmod r_i \in \{0,1,2\}$; that is, each disc in C^i intersects two discs ahead and two discs behind it in the cycle, but no other disc in C^i. Discs corresponding to u_i should not intersect discs corresponding to any other variable u_j (see Figure 6.6). That is,

$$\forall i,j,\ \forall C^i_x \in C^i,\ \forall C^j_y \in C^j,\ i \neq j \Rightarrow C^i_x \cap C^j_y = \varnothing.$$

A simple counting argument shows that if $|C^i| = 3m_i$, then at least $2m_i$ points are necessary to 2-stab C^i.

We now proceed to describe in more detail the structure of possible 2-stabbing sets of C^i that use exactly $2m_i$ points. In particular, we explain how these sets correspond to truth values of variables in the reduction from Planar 3-SAT.

Observation 6.3.1. *If F' is a multiset of $2m_i$ points that 2-stabs C^i, then each $q \in F'$ must stab three discs in C_i; furthermore, $q \in C^i_j \cap C^i_{j+1} \cap C^i_{j+2}$ for some j.*

Lemma 6.3.1. *If F' is a multiset of $2m_i$ points that 2-stabs C^i, then*

$$\forall j \in \{0,1,...,r_{i-1}\},\ |C^i_j \cap F'| = 2.$$

Proof. By Observation 6.3.1, each point in F' stabs three discs in C^i. Since $|F'| = 2m_i$, there are $6m_i$ discs (counting multiplicities) stabbed by F'. Let D denote the multiset of discs stabbed by F'. Since F' is a 2-stabber of C^i, each disc of C^i must occur at least twice in D. As $|D|/|C^i| = 2$, we conclude that each disc of C^i appears exactly twice in D. That is, each disc of C^i is stabbed by exactly

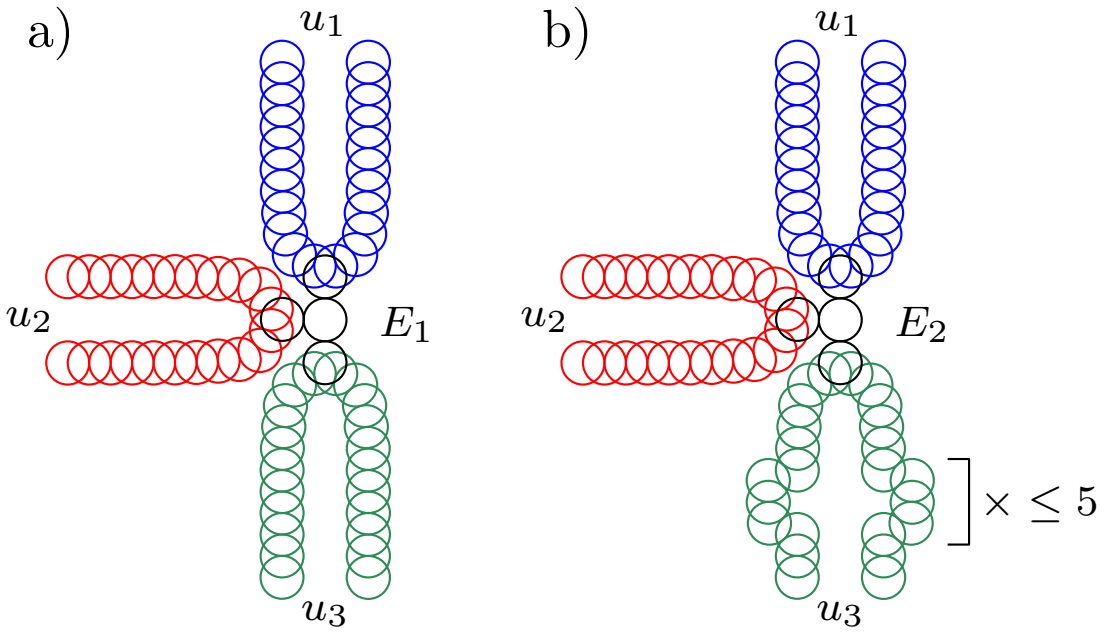

Figure 6.3: **Clause Gadget.** Each clause E_i is represented by a central disc tangent to three adjacent discs (black), each of which intersects the chain of discs corresponding to one of the three variables in the clause. This example illustrates two clauses: $E_1 = u_1 \vee u_2 \vee u_3$ and $E_2 = u_1 \vee u_2 \vee \neg u_3$. To negate the truth value of a variable, additional discs are added to the variable chain on either side of the clause gadget to switch the truth value of the variable where it intersects the black disc. In this example, one additional disc is added to each side of the green chain to negate u_3 in E_2. Since the number of discs in each variable disc chain must be a multiple of 6, additional discs may be required on the other side of the intersection with the black disc.

two points of F'. □

With Observation 6.3.1 and Lemma 6.3.1, Lemmas 6.3.2 and 6.3.3 below show that there are six combinatorially distinct configurations of $2m_i$ points that 2-stab C^i: three cases correspond to the truth value *True*, and the other three correspond to the truth value *False* (see Figures 6.1 and 6.2).

Lemma 6.3.2. *Suppose F' 2-stabs C^i and $|F'| = 2m_i$. Then for every j, $q_j \cap F' = \varnothing$, $q_{j+1} \cap F' = \varnothing$, or $q_{j+2} \cap F' = \varnothing$, where*

$$q_x \in C^i_x \cap C^i_{x+1} \cap C^i_{x+2}.$$

Lemma 6.3.2 implies that F' cannot contain three points that lie on consecutive intersections of three discs.

Proof. Suppose there is a j such that $\{q_j, q_{j+1}, q_{j+2}\} \subseteq F'$. Without loss of generality, assume $j = 0$. In this case, $|C^i_2 \cap F'| = 3$, contradicting Lemma 6.3.1. □

Lemma 6.3.3. *Suppose F' 2-stabs C^i, $|F'| = 2m_i$, and at least one point occurs twice in F'. Then every point occurs twice in F'.*

Proof. Suppose some point q_j occurs twice in F' and some point q_k occurs only once in F'. Without loss of generality, suppose $j = 0$, $k > 0$, and $k - j$ is minimum among all such q_j and q_k, where $q_x \in C^i_x \cap C^i_{x+1} \cap C^i_{x+2}$. By Lemma 6.3.1, $k \notin \{1, 2\}$. In order to stab the disc C^i_3, F' must contain q_3. Therefore, $k = 3$ and, by assumption, q_3 occurs only once in F'. Consequently, F' cannot 2-stab the disc C^i_3, contradicting the assumption that F' 2-stabs C^i. □

We now characterize all possible configurations F' of $2m_i$ points that 2-stab C^i. If F' contains some point twice, then by Lemma 6.3.3 F' has m distinct points, each of which appears twice in F'. By Lemma 6.3.1, F' must be the multiset $\{\{q_{0+j}, q_{0+j}, q_{3+j}, q_{3+j}, ..., q_{r_i-3+j}, q_{r_i-3+j}\}\}$, for some $j \in \{0, 1, 2\}$, corresponding

to Figures 6.1(b), 6.1(c), and 6.2(a), respectively. In our reduction, we associate the 2-stabbing set for $j = 0, 1$ with *True* and the 2-stabbing set for $j = 2$ with *False*.

The only other possibility is that F' contains $2m$ distinct points. In this case, Lemmas 6.3.1 and 6.3.2 imply that F' is the set $\{q_{0+j}, q_{1+j}, q_{3+j}, q_{4+j}, ..., q_{r_i-3+j}, q_{r_i-2+j}\}$, for some $j \in \{0, 1, 2\}$, corresponding to Figures 6.1(a), 6.2(b), and 6.2(c), respectively. In our reduction, we associate the 2-stabbing set for $j = 0$ with *True* and the 2-stabbing set for $j = 1, 2$ with *False*.

Consequently, exactly six configurations are possible for F'. We further restrict the possible configurations by adding three *constraint discs* to each literal's chain of discs, as illustrated in Figure 6.4. In three cases, two of the constraint discs contain one or two points that stab C^i, and two additional points are necessary and sufficient to 2-stab all three constraint discs. In the remaining three cases, all three constraint discs are empty and cannot be 2-stabbed by any two additional points. Therefore, this reduces the number of possible configurations to three. Position the three constraint discs such that Figures 6.4c, e, and b correspond to Figures 6.1a, 6.1b, and 6.2c, respectively. This allows negating a variable before it meets a clause gadget (if the variable is negated in that clause) by adding discs (see Figure 6.3).

6.3.1 Clause Gadget and Reduction

We now describe the clause gadgets in our reduction. Each clause $E_h \in E$ is represented by four discs: one central disc that intersects three other discs, each in a point (see Figures 6.3 and 6.5). If the clause contains the literal u_i, then the corresponding clause gadget intersect a disc C^i_j such that $j \bmod 3 \in \{0, 1\}$; if the clause contains the literal $\neg u_i$, then $j \bmod 3 = 2$. The index j of the intersecting disc can be controlled by adding discs (see Figure 6.3). If the assignment of a truth value to u_i implies that some point $q \in C^i_{j-1} \cap C^i_j \cap C^i_{j+1}$ is selected for the

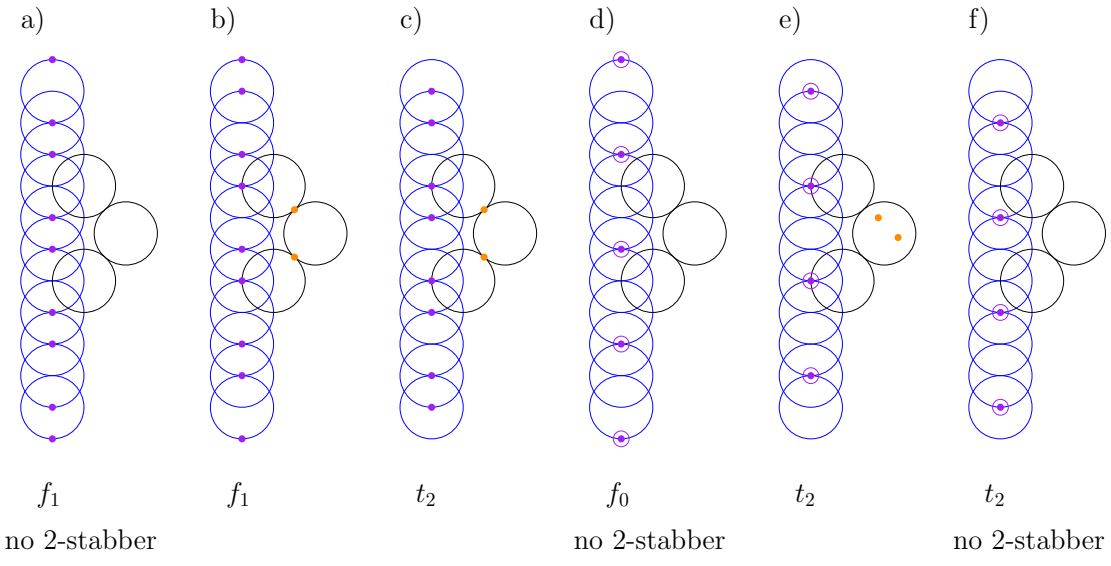

Figure 6.4: Adding three constraint discs (black) to each variable gadget reduces the number of possible configurations from six to three; the configurations b, c, and e can each be 2-stabbed with two additional points (orange), but a, d, and f cannot.

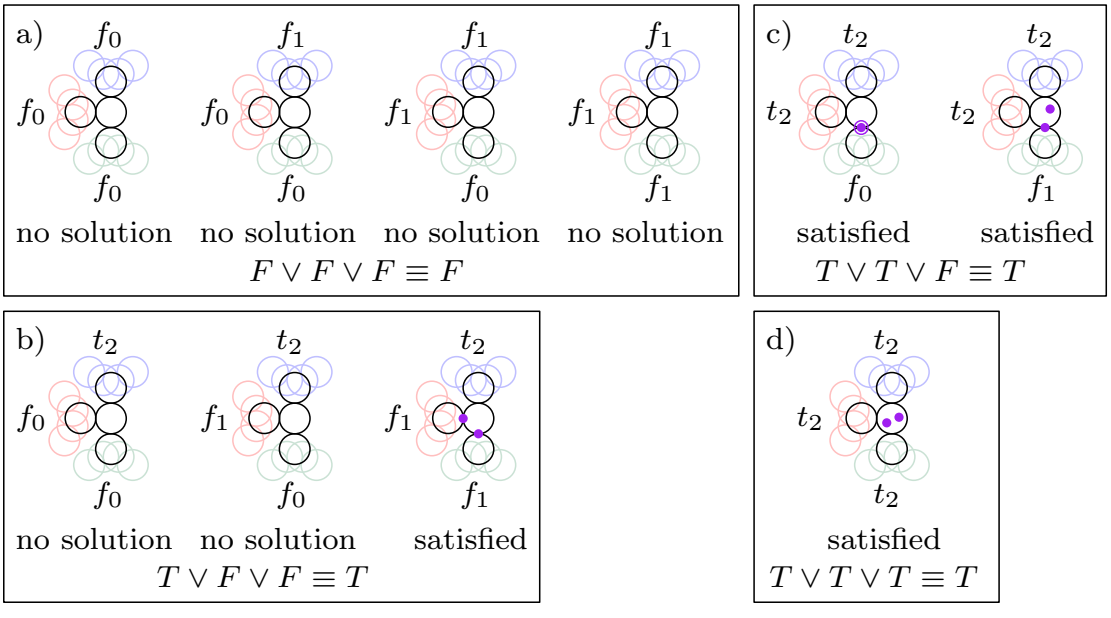

Figure 6.5: **Clause gadget configurations.** The black discs labelled f_0 and f_1 corresponds to false variables, and contain zero or one points, respectively. The black discs labelled t_2 correspond to true variables, and contain two points. Two additional points (purple) suffice to 2-stab clause gadgets that have one or more true literals (b–d). No configuration of two points can 2-stab a clause gadget with three false literals (a). Two purple points are collocated in the left configuration of (c).

solution F, then q may be selected so as to belong to the black disc as well. These correspond exactly to configurations of points for true literals (see Figure 6.1 and 6.2).

Observation 6.3.2. *A clause gadget can be 2-stabbed by two points if and only if at least one of its literals is true. Furthermore, at least two points are required to 2-stab the clause gadget for every combination of truth assignments to its literals.*

Let Ψ denote the set $\bigcup_{i=1}^{n} C^i$ of discs in variable gadgets, along with the $4s$ discs for the s clauses in E, and the $3n$ constraint discs for the n variables in U. Let

$$k = 2s + 2n + 2\sum_{i=1}^{n} m_i. \tag{6.3.1}$$

We now prove that E is satisfiable if and only if there exists a set F of k points that 2-stabs Ψ.

First, assume that E is satisfied by a truth assignment Γ to U. For $i \in \{1, \ldots, n\}$, if $\Gamma(u_i) = \mathit{True}$, then add to F $2m_i$ points that 2-stab C^i as in Figure 6.1a. Otherwise, $\Gamma(u_i) = \mathit{False}$; add to F $2m_i$ points as in Figure 6.2b. So far, $|F| = 2\sum_{i=1}^{n} m_i$, and each set C^i of discs is 2-stabbed by F. By our assumption, each clause $E_h = u_x \vee u_y \vee u_z$ is satisfied; add to F two points per clause as in Figure 6.5 such that each clause's set of four discs is 2-stabbed by F. Finally, add to F two points per variable as in Figure 6.4, such that each variable's constraint discs are 2-stabbed by F. Therefore, $|F| = k$ and F 2-stabs Ψ.

To prove the converse, assume F is a set of k points that 2-stabs Ψ. We must show there exists a truth assignment Γ for U that satisfies E. By Observation 6.3.2, at least $2s$ points of F are required to 2-stab the clause gadgets and at least $2n$ points of F are required to 2-stab the constraint discs for variable gadgets (excluding points that 2-stab $\bigcup_{i=1}^{n} C^i$) by (6.3.1); this leaves at most $2\sum_{i=1}^{n} m_i$ points in F to 2-stab $\bigcup_{i=1}^{n} C^i$. By Observation 6.3.1 and Lemmas 6.3.2 and 6.3.3, $2m_i$ points are necessary in F to 2-stab each C^i; furthermore, these correspond to

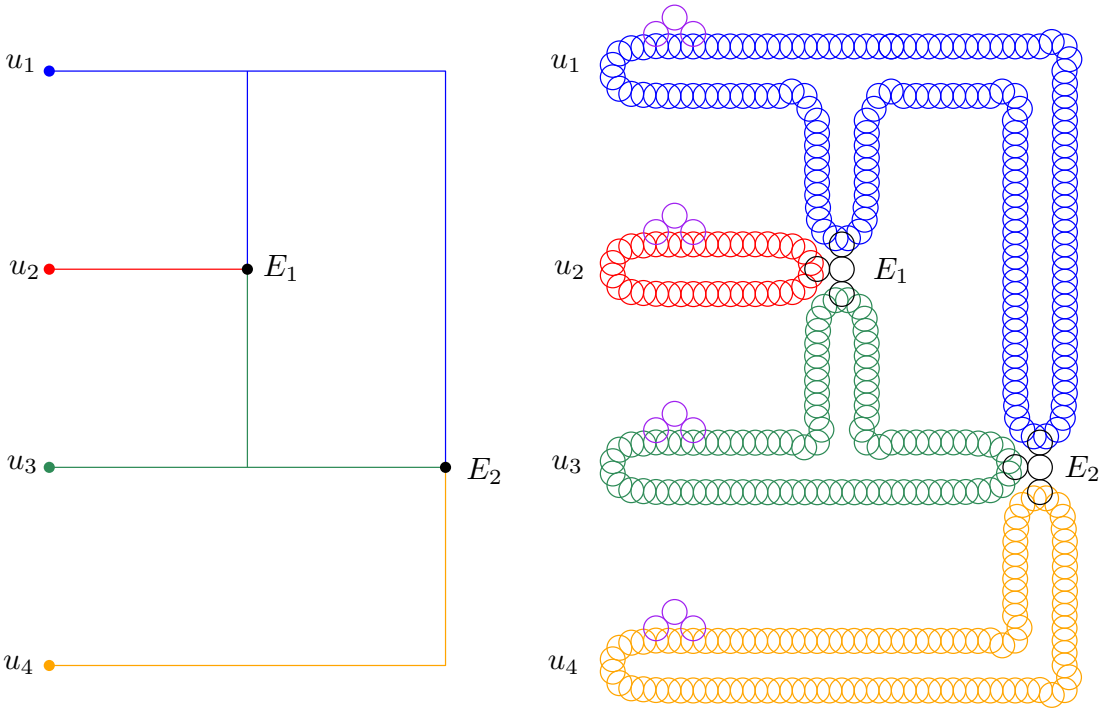

Figure 6.6: Given a rectilinear planar embedding of G_ϕ (left), we replace the edges adjacent to each variable in U with a cycle of overlapping discs (right) and each clause in E with four discs (black) that overlap the three cycles of discs corresponding to variables in that clause. Each variable also has one set of three constraint discs (purple). This example illustrates the reduction for an instance consisting of the clauses $E_1 = u_1 \vee u_2 \vee u_3$ and $E_2 = u_1 \vee u_3 \vee u_4$.

one of the configurations in Figures 6.1 or 6.2. That is, for each C^i, F uniquely determines the truth assignment $\Gamma(u_i)$ associated with that point configuration. By (6.3.1), this leaves

$$k - 2\sum_{i=1}^{n} m_i = 2s + 2n,$$

points with which to 2-stab the s clause gadgets (Figures 6.3 and 6.5) and the n sets of variable constraint discs (Figure 6.4). Since each clause gadget is 2-stabbed by F, by Observation 6.3.2, each clause in E is satisfied. See Figure 6.6 for an example of the reduction. Given any set P' of unit discs and any set F of points in $\mathbb{R}^2$, it is straightforward to verify in polynomial time whether F 2-stabs P'.

Theorem 6.3.4. *Disc 2-Stabbing is NP-complete.*

Since every P' and F in $\mathbb{R}^2$ can be embedded into $\mathbb{R}^d$ for any $d \geq 2$, this gives the following theorem:

Theorem 6.3.5. *For all $d \geq 2$, the ℓ-fault-tolerant k-center problem is NP-complete in $\mathbb{R}^d$.*

In this chapter, we proved that the ℓ-fault-tolerant Euclidean k-center problem is NP-hard. Our proof was based on a reduction from the well-known Planar 3-SAT problem to the disc 2-stabbing problem. The disc 2-stabbing problem asks whether, given a collection of discs in the plane, there exist a multiset of of size k points such that every disc contains at least two of these k points. We showed how an instance of Planar 3-SAT can be transformed into an instance of disc 2-stabbing, and then did a reduction to the ℓ-fault-tolerant k-center decision problem.

This reduction demonstrates that solving the ℓ-fault-tolerant Euclidean k-center problem efficiently in polynomial time would also allow us to solve Planar 3-SAT in polynomial time, which is believed to be impossible. Therefore, our result establishes the computational hardness of the problem and explains why no efficient exact algorithm is expected to exist in higher dimensions.

Chapter 7

2D Approximation Algorithm

In this chapter, we present a simple 2-approximation algorithm for the fault-tolerant Euclidean k-center problem. Our method builds upon the well-known greedy k-center approximation algorithm originally proposed by Gonzalez [15].

7.1 k-Center Approximation Algorithm of Gonzalez

The key idea of Gonzalez's algorithm is to iteratively select centers that are as far apart as possible, ensuring that each new center covers the largest remaining uncovered distance. By adapting this greedy strategy to the fault-tolerant setting, we obtain a straightforward algorithm that guarantees a 2 factor approximation of the optimal solution.

Algorithm 1: Greedy 2-approximation for the ℓ-fault-tolerant Euclidean k-center problem.

Input : A set $P \subset \mathbb{R}^2$ of n points, integers k, ℓ satisfying $0 < \ell \leq k < n$.

Output: A multiset of (at most) k points from P.

1 $k' \leftarrow \lfloor k/\ell \rfloor$

2 Select an arbitrary point $s_1 \in P$

3 $S \leftarrow \{s_1\}$

4 $i \leftarrow 1$

5 **while** $|S| < k'$ **do**

6 $\quad s_{i+1} \leftarrow \arg\max_{p \in P \setminus S} d(p, S)$

7 $\quad r_i \leftarrow d(s_{i+1}, S)$

8 $\quad S \leftarrow S \cup \{s_{i+1}\}$

9 $\quad i \leftarrow i + 1$

10 **end**

11 **return** ℓ copies of each point in S

When $\ell = 1$, Algorithm 1 is identical to the greedy 2-approximation algorithm of Gonzalez [15], which in each iteration locates a facility that is farthest from the facilities already selected. Line 7 of the algorithm is used only for the analysis of the algorithm.

Theorem 7.1.1. *Algorithm 1 is a 2-approximation for the ℓ-fault-tolerant Euclidean k-center problem.*

Proof. Let $k' \leftarrow \lfloor k/\ell \rfloor$ and $S = \{s_1, s_2, ..., s_{k'}\}$ be the k' centers selected by Lines 1–10 of Algorithm 1, where s_i is the i^{th} facility selected. The algorithm returns ℓ copies of S and, therefore, the cost of the solution returned is equal to the cost of the Euclidean k'-center solution S.

Let $r_{k'} = \text{cost}_1(P, S)$ and let $s_{k'+1}$ be a point in P that realizes $r_{k'}$. The value $r_{k'}$ is the cost of the solution returned by (Line 11) of the algorithm as each point

in S appears ℓ times in the solution. Since $k < n$, we have $k' < n$ and, therefore, the point $s_{k'+1}$ is distinct from the points in S.

The values $r_1, \ldots, r_{k'-1}$ computed in Line 7 and $r_{k'}$ defined above satisfy $r_1 \geq \cdots \geq r_{k'-1} \geq r_{k'}$. These inequalities imply that any two points in $S \cup \{s_{k'+1}\}$ are at least a distance of $r_{k'}$ apart.

Let OPT denote the cost of an optimal solution for the ℓ-fault-tolerant k-center instance. It suffices to show that $r_{k'} \leq 2 \cdot OPT$, which we show by contradiction.

Suppose $r_{k'} > 2 \cdot OPT$ and let $\mathcal{O}$ be an optimal solution, where $|\mathcal{O}| = k$. Consider any point $p \in S \cup \{s_{k'+1}\}$. As $p \in P$, there must be at least ℓ facilities in $\mathcal{O}$ that are within distance OPT from p. Since $r_{k'} > 2 \cdot OPT$ and by the triangle inequality, none of these facilities are within a distance of OPT from any member of $S \cup \{s_{k'+1}\} \setminus \{p\}$. This implies that there must be at least $(k'+1)\ell > k$ facilities in $\mathcal{O}$, which is a contradiction and $r_{k'} \leq 2 \cdot OPT$. □

The running time of Algorithm 1 is mainly determined by the loop structure. The algorithm first initializes by selecting an arbitrary point and setting it as the first center. After that, the main work happens in the **while loop** (Lines 5–8). This loop continues until k' centers have been selected. Therefore, the outer loop runs exactly k' times.

Inside each iteration of the loop, the algorithm must find the point s_{i+1} that is farthest from the current set of chosen centers S. To do this, the algorithm compares the distance of every remaining point in P to S. Since there are at most n points to check, this step takes $\Theta(n)$ time in each iteration.

Thus, the cost of one loop iteration is $\Theta(n)$, and because the loop repeats k' times, the total running time is $\Theta(nk')$. Substituting $k' = \lfloor k/\ell \rfloor$, we obtain the overall running time of $\Theta(nk/\ell)$. This bound shows that the greedy algorithm remains efficient even in the fault-tolerant version.

Figure 7.2 illustrates an example that demonstrates the limitation of the algorithm: in the worst case, Algorithm 1 cannot achieve an approximation factor smaller than 2. This shows that the 2-approximation bound is tight.

Finally, Theorem 7.1.1 establishes that this 2-approximation guarantee holds not only for the Euclidean version of the problem but also more generally. In fact, the result applies to both the discrete and continuous variants of the ℓ-fault-tolerant k-center problem, and it remains valid in any metric space.

7.1.1 Illustrative Example of Algorithm 1

Figure 7.1 shows the result of applying Algorithm 1 to a set of 18 client points on the plane. The greedy algorithm was used to solve the 2-fault-tolerant 6-center problem. In the figure, the three red crosses are the centers chosen by the algorithm, and since $\ell = 2$, each of them is doubled to make six centers in total. The blue circles represent the exact optimal solution for the 2-fault-tolerant 6-center problem. In this example, the greedy algorithm gives a covering radius of approximately 7.21, while the optimal radius is 4.

This gap between 7.21 and 4 demonstrates that the greedy algorithm does not always achieve the optimal solution. More importantly, it provides an explicit instance that serves as a lower bound on the approximation factor of Algorithm 1. That is, this example proves that the algorithm can output a solution whose cost is at least $\frac{7.21}{4} \approx 1.8$ times larger than the optimal. Note that this example is just a random instance; in contrast, we also provide another Figure 7.2 that demonstrates a case where the algorithm achieves a 2-approximation, showing that the algorithm can be tight for certain instances. Hence, the performance guarantee of the greedy algorithm cannot be improved beyond this ratio.

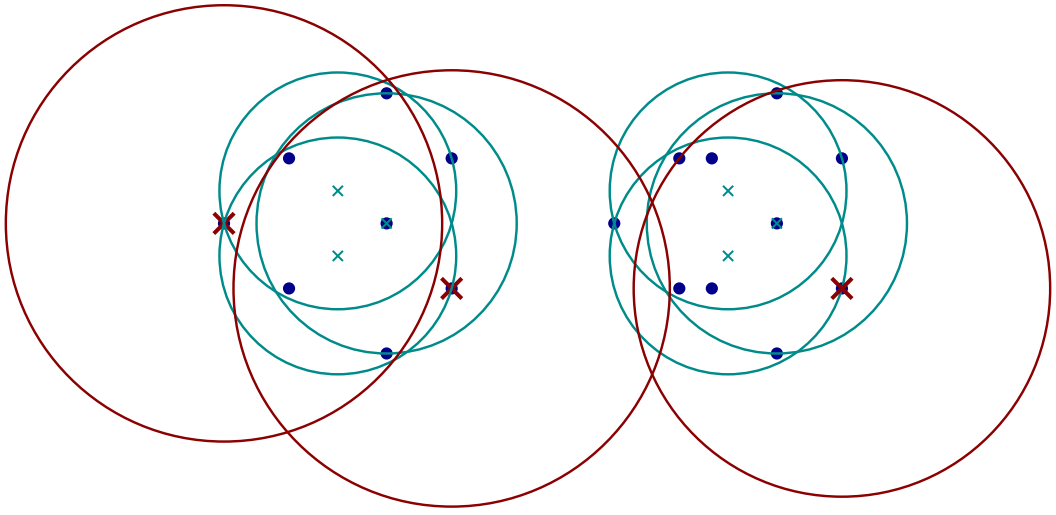

Figure 7.1: Given this set of dark blue client points •, the exact 2-fault-tolerant 6-center corresponds to centers of the six light-blue circles, and × denote the approximate 2-fault-tolerant 6-center returned by Algorithm 1, with two centers placed on each of the three points marked × (these coincide with blue input points).

7.2 Tightness Discussion

In the classical k-center problem, Gonzalez's [15] greedy algorithm is known to achieve a 2-approximation, and this bound is tight. On the one hand, worst-case examples are known in which the algorithm's output has cost exactly twice the optimum, showing that the 2-approximation analysis cannot be improved for the algorithm itself. On the other hand, it has also been shown that [10], unless NP=P, no polynomial-time algorithm can achieve an approximation factor strictly better than 2 for discrete k-center and better than $\frac{1+\sqrt{7}}{2} \approx 1.8229$ for Euclidean non-fault tolerant k-center.

The extension of Gonzalez's algorithm to approximate the solution for the Euclidean fault-tolerant k-center also guarantees a 2-approximation, and there exist instances (see Figure 7.2), where the cost of the returned solution is exactly twice the optimal cost. This shows that the analysis of the algorithm is tight: its performance guarantee cannot be improved below 2. Thus, the analysis of the Gonzalez's algorithm is tight for both discrete and Euclidean k-center.

However, it is not known whether 2 is the best possible approximation factor for

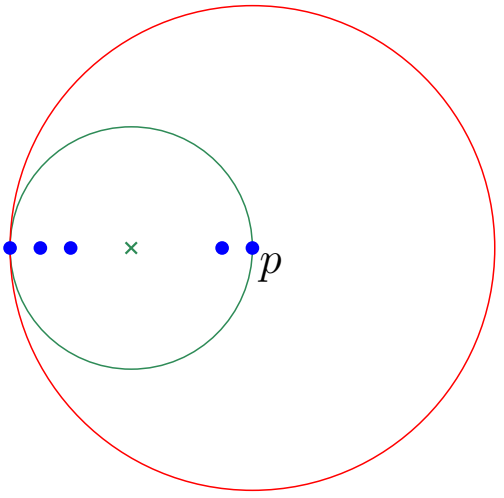

Figure 7.2: In this example, the unique optimal 2-fault-tolerant 2-center has two facilities on $\times$, whereas Algorithm 1 could place two facilities on p.

the ℓ-fault-tolerant k-center problem. In fact, other algorithmic approaches might be capable of achieving approximation ratios below 2 for certain settings.

Can Algorithm 1 or its analysis be improved to get a better factor than 2-approximation? No. Is the 2-approximation bound necessarily tight for the problem itself? We don't know!

In this chapter, we presented a simple and practical 2-approximation algorithm for the fault-tolerant Euclidean k-center problem. We described the algorithm, proved that it achieves a 2-approximation, and analyzed its running time, which is $\Theta(nk/\ell)$, making it efficient for large instances. Using examples and illustrations, we showed that the 2-approximation bound is tight for the algorithm itself. At the same time, it remains an open question whether the fault-tolerant k-center problem admits a polynomial-time algorithm with an approximation factor strictly better than 2.

Chapter 8

Conclusion and Future Work

In this thesis, we studied the ℓ-fault-tolerant Euclidean k-center problem. We started by giving the necessary background and definitions, and then reviewed the main related work from both classical and fault-tolerant settings.

We looked at the geometric properties of the problem and showed how the k-center problem can be solved in one dimension, even when fault tolerance is required.

In two dimensions, we proved that the fault-tolerant Euclidean k-center problem is NP-hard. Because of this, we focused on approximation algorithms. We proposed a 2-approximation algorithm to the fault-tolerant case and showed that it still achieves a factor of 2. We also discussed its running time and limits.

These results give a first step toward understanding the complexity of ℓ-fault-tolerant Euclidean k-center problem. However, they leave us with several open directions. In the following sections, we explain some of these open directions.

8.1 Approximation by a $\lfloor k/\ell \rfloor$-Center

Our example in Figure 4.4 demonstrates that the straightforward strategy of placing ℓ points on each facility of a Euclidean $\lfloor k/\ell \rfloor$-center for a point set P cannot

guarantee an approximation factor better than $(1/\sqrt{3} + 1/2) \approx 1.0774$ in the general case. This result shows that even simple and natural strategies can have inherent limitations when applied to the ℓ-fault-tolerant Euclidean k-center problem.

It may be possible to improve this lower bound by studying other examples in more detail. For instance, Drezner's configuration shown in Figure 4.3 might lead to a stronger hardness result. In particular, if one could compute the Euclidean 2-center for that arrangement of points, the analysis might yield a larger lower bound on the approximation factor.

On the positive side, Theorem 7.1.1 guarantees that a 2-approximation is always possible. This means we can always find a solution whose maximum radius is at most twice the optimal. However, it is still an open problem to determine the best achievable approximation factor in the range $[1/\sqrt{3} + 1/2, 2)$. In other words, we do not yet know whether there exists a polynomial time algorithm that can always achieve an approximation strictly between about 1.0774 and 2.

The running time of our 2-approximation algorithm is $O(nk/\ell)$, which is polynomial and efficient in practice. Nevertheless, there may be ways to improve its running time further. In particular, it might be possible to adapt the techniques developed by Feder and Greene [10] to reduce the running time to $O(n \log k)$. Designing such an improved algorithm remains an interesting and important direction for future work.

8.2 Hardness of Approximation

We have shown that the ℓ-fault-tolerant Euclidean k-center problem becomes NP-hard when the input points are in two or more dimensions. This means that there is no known algorithm that can solve this problem exactly in polynomial time, unless P = NP.

Feder and Greene [10] proved that even for the standard (non-fault-tolerant) Euclidean k-center problem, it is NP-hard to compute a solution that is within a factor of $\frac{1+\sqrt{7}}{2} \approx 1.8229$ of the optimal solution, when the problem is in two or more dimensions. This means that no polynomial-time algorithm can guarantee a solution that is better than this approximation factor, unless P = NP.

For the ℓ-fault-tolerant version of the problem, a similar result is expected, but has not been proven yet. That is, it is still an open question whether it is NP-hard to approximate the fault-tolerant Euclidean k-center problem for $\ell \geq 2$ within a factor better than $\frac{1+\sqrt{7}}{2}$.

In fact, even for the standard (non-fault-tolerant) Euclidean k-center problem in two dimensions, it is still unknown whether there exists a polynomial-time algorithm that can guarantee an approximation factor α for any α strictly between $\frac{1+\sqrt{7}}{2}$ and 2.

On the other hand, this kind of difficulty does not appear in the *discrete* k-center problem. For the discrete version, there is a well-known algorithm that gives a 2-approximation in polynomial time [15]. Furthermore, it has been proven that it is NP-hard to find a better approximation factor than 2; that is, finding a $(2-\epsilon)$-approximation for any $\epsilon > 0$ is NP-hard.

8.3 Minimum Distance Constraint

In the ℓ-fault-tolerant Euclidean k-center problem, an optimal solution may collocate multiple facilities at the same location. This is allowed in the mathematical model, but in practice such collocation is often unrealistic. For example, in applications such as EV charging networks, cloud servers, or emergency service placement, it is not practical to build two identical facilities at exactly the same point.

This motivates a new variant of the problem. Suppose we require that every pair of facilities must be separated by at least a fixed minimum distance $\delta > 0$. Then the objective is to find k facility locations that satisfy both the fault-tolerant coverage property and the spacing constraint. Formally, the problem can be defined as follows:

Definition 8.3.1 (ℓ-Fault-Tolerant Euclidean k-Center with Minimum Distance Constraint)**.** *Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points in $\mathbb{R}^d$, integers k, ℓ such that $1 \leq \ell \leq k$, and a minimum distance parameter $\delta > 0$, find a set $F = \{f_1, f_2, \ldots, f_k\} \subseteq \mathbb{R}^d$ such that, for every $p \in P$, the distance to its ℓ-th nearest center in F gets minimized, i.e.,*

$$\min_{\substack{F \subseteq \mathbb{R}^d \\ |F|=k}} \max_{p \in P} \operatorname{dist}_\ell(p, F),$$

subjected to an extra constraint:

$$\textit{For every distinct} \quad f_i, f_j \in F, \quad \textit{it holds that} \quad \|f_i - f_j\|_2 \geq \delta,$$

Recall $\operatorname{dist}_\ell(p, F)$ *is the distance from p to its ℓ-th nearest facility.*

Adding this minimum distance constraint changes the nature of the problem substantially. In the classical setting, duplicating each facility ℓ times already ensures ℓ-fault tolerance, but this construction becomes invalid under the spacing requirement. The problem is now closely related to geometric packing and placement problems, which are known to be computationally hard.

Therefore, the approximability of the ℓ-fault-tolerant Euclidean k-center problem under a minimum distance constraint is an open question. It is not known whether constant-factor approximation algorithms can be achieved, or if this version admits stronger inapproximability results.

Appendix A

Appendix

Algorithm 2: Gonzalez's Greedy Algorithm for the k-Center Problem

Input : A set P of n points in a metric space $(\mathcal{M}, \text{dist})$, integer k.

Output: A set F of k centers.

Select an arbitrary point $p_1 \in P$ as the first center.

$F \leftarrow \{p_1\}$

while $|F| < k$ **do**

    Select $p \in P$ that maximizes $\text{dist}(p, F)$.

    `// (the farthest point from the current set of centers)`

    Add p to F.

end

return F

Bibliography

[1] H. H. Ali and L. E. Kadhum, *K-means Clustering Algorithm Applications in Data Mining and Pattern Recognition*, International Journal of Science and Research (IJSR) **6** (2017), no. 8, 1577–1584.

[2] T. M. Chan, *More Planar Two-Center Algorithms*, Computational Geometry **13** (1999), no. 3, 189–198.

[3] Sh. Chaudhuri, N. Garg, and R. Ravi, *The p-Neighbor k-Center Problem*, Information Processing Letters **65** (1998), no. 3, 131–134.

[4] Sh. Chechik and D. Peleg, *The Fault-Tolerant Capacitated k-Center Problem*, Theoretical Computer Science **566** (2015), 12–25.

[5] D. Z. Chen, J. Li, and H. Wang, *Efficient Algorithms for the One-dimensional k-Center Problem*, Theoretical Computer Science **592** (2015), 135–142.

[6] K. Cho, E. Oh, H. Wang, and J. Xue, *Optimal Algorithm for the Planar Two-Center Problem*, 40th International Symposium on Computational Geometry (SoCG) (2024), 40:1–40:15.

[7] R. Cole, *Slowing Down Sorting Networks to Obtain Faster Sorting Algorithms*, Journal of the ACM (JACM) **34** (1987), no. 1, 200–208.

[8] Z. Drezner, *Heuristic Solution Methods for Two Location Problems with Unreliable Facilities*, Journal of the Operational Research Society (JORS) **38** (1987), no. 6, 509–514.

[9] D. Eppstein, *Faster Construction of Planar Two-Centers*, Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (1997), 131–138.

[10] T. Feder and D. Greene, *Optimal Algorithms for Approximate Clustering*, Proceedings of the 20th Annual ACM Symposium on Theory of Computing, 1988, pp. 434–444.

[11] C. G. Fernandes, S. P. de Paula, and L. L. Pedrosa, *Improved Approximation Algorithms for Capacitated Fault-Tolerant k-Center*, Algorithmica **80** (2018), no. 3, 1041–1072.

[12] H. Fournier and A. Vigneron, *Fitting a Step Function to a Point Set*, European Symposium on Algorithms (ESA 2008) **5193** (2008), 442–453.

[13] G. N. Frederickson, *Parametric Search and Locating Supply Centers in Trees*, Workshop on Algorithms and Data Structures (WADS), 1991, pp. 299–319.

[14] M. R. Garey and D. S. Johnson, *Computers and intractability*, W. H. Freeman New York, 1979.

[15] T. F. Gonzalez, *Clustering to Minimize the Maximum Intercluster Distance*, Theoretical Computer Science **38** (1985), 293–306.

[16] D. S. Hochbaum, *When Are NP-hard Location Problems Easy?*, Annals of Operations Research **1** (1984), no. 3, 201–214.

[17] D. S. Hochbaum and D. B. Shmoys, *A Best Possible Heuristic for the k-Center Problem*, Mathematics of Operations Research **10** (1985), no. 2, 180–184.

[18] R. Huang, S. Kim, and M.B.C Menezes, *Facility Location for Large-Scale Emergencies*, Annals of Operations Research **181** (2010), 271–286.

[19] J. W. Jaromczyk and M. Kowaluk, *An Efficient Algorithm for the Euclidean Two-Center Problem*, Proceedings of the 10th Annual Symposium on Computational Geometry (SoCG), 1994, pp. 303–311.

[20] M. Jeger and O. Kariv, *Algorithms for Finding p-Centers on a Weighted Tree (for relatively small p)*, Networks **15** (1985), no. 3, 381–389.

[21] H. Jia, F. Ordóñez, and M. Dessouky, *A Modeling Framework for Facility Location of Medical Services for Large-scale Emergencies*, IISE Transactions **39** (2007), no. 1, 41–55.

[22] O. Kariv and S. L. Hakimi, *An algorithmic approach to network location problems. ii: The p-medians*, SIAM Journal on Applied Mathematics **37** (1979), no. 3, 539–560.

[23] ———, *An Algorithmic Approach to Network Location Problems. I: The p-Centers*, SIAM Journal on Applied Mathematics **37** (1979), no. 3, 513–538.

[24] S. Khuller, R. Pless, and Y. J. Sussmann, *Fault Tolerant k-Center Problems*, Theoretical Computer Science **242** (2000), no. 1-2, 237–245.

[25] S. Khuller and Y. J. Sussmann, *The Capacitated k-Center Problem*, SIAM Journal on Discrete Mathematics **13** (2000), no. 3, 403–418.

[26] D. E. Knuth and A. Raghunathan, *The Problem of Compatible Representatives*, SIAM Journal on Discrete Mathematics **5** (1992), 422–427.

[27] S. O. Krumke, *On a Generalization of the p-Center Problem*, Information Processing Letters **56** (1995), 67–71.

[28] N. Kumar and B. Raichel, *Fault Tolerant Clustering Revisited*, Proc. Canadian Conference on Computational Geometry (CCCG), 2013, pp. 103–108.

[29] X. Lai, *New Algorithms for Two Logistics Optimization Problems* (2015).

[30] D. Lichtenstein, *Planar Formulae and Their Uses*, SIAM Journal on Computing **11** (1982), no. 2, 329–343.

[31] N. Megiddo, *Linear-time Algorithms for Linear Programming in R^3 and Related Problems*, SIAM Journal on Computing **12** (1983), no. 4, 759–776.

[32] ———, *Linear Programming in Linear Time When the Dimension is Fixed*, Journal of the ACM (JACM) **31** (1984), no. 1, 114–127.

[33] N. Megiddo and K. J. Supowit, *On the Complexity of Some Common Geometric Location Problems*, SIAM Journal on Computing **13** (1984), no. 1, 182–196.

[34] N. Megiddo and A. Tamir, *New Results on the Complexity of p-Center Problems*, SIAM Journal on Computing **12** (1983), no. 4, 751–758.

[35] M. T. Melo, S. Nickel, and F. Saldanha-Da-Gama, *Facility Location and Supply Chain Management–A Review*, European Journal of Operational Research **196** (2009), no. 2, 401–412.

[36] P. B. Mirchandani and R.L. Francis, *Discrete Location Theory*, 1990.

[37] M. Sharir and D. Eppstein, *A Near-Linear Algorithm for the Planar 2-Center Problem*, Discrete & Computational Geometry **18** (1997), 125–134.

[38] D. B. Shmoys, *The Design and Analysis of Approximation Algorithms: Facility Location as a Case Study*, Trends in Optimization, AMS Proceedings of Symposia in Applied Mathematics, 2004, pp. 85–97.

[39] H. Wang, *On the Planar Two-Center Problem and Circular Hulls*, Discrete & Computational Geometry **68** (2022), no. 4, 1175–1226.

[40] E. Welzl, *Smallest Enclosing Disks (Balls and Ellipsoids)*, New Results and New Trends in Computer Science, 1991, pp. 359–370.