REPUBLIC OF TURKEY
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF INDUSTRIAL ENGINEERING

Obnoxious Facility Location Models and Solution Algorithms

YELİZ YILERİ TERKEŞLİ

MASTER OF SCIENCE

ADANA 2023

REPUBLIC OF TURKEY
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF INDUSTRIAL ENGINEERING

Obnoxious Facility Location Models and Solution Algorithms

YELİZ YILERİ TERKEŞLİ

M.Sc.

SUPERVISOR
ASST. PROF. DR. NUŞİN UNCU
CO-SUPERVISOR
ASSOC. PROF. DR. FATİH KILIÇ

ADANA, 2023

DECLARATION OF CONFORMITY

In this thesis study, which was prepared following the thesis writing rules of Adana Alparslan Türkeş Science and Technology University Institute of Graduate School, I declare that I provide all the information, documents, evaluations and results in accordance with scientific ethics and moral codes without resorting to any means or assistance that would be contrary to scientific ethics and traditions. I also declare that I refer to all of the articles I used in this study with appropriate references and accept all moral and legal consequences if a situation is found contrary to my statement regarding my work.

16/08/2023

[Signature]

YELİZ YILERİ TERKEŞLİ

ÖZET

Obnoxious Facility Location Models and Solution Algorithms

YELİZ YILERİ TERKEŞLİ
Yüksek Lisans, Endüstri Mühendisliği Anabilim Dalı
Danışman: Dr. Öğr. Üyesi Nuşin UNCU
II. Danışman: Doç. Dr. Fatih KILIÇ
Ağustos 2023, 59 sayfa

Bu tezde, zararlı tesislerin topluluklar ve çevre üzerindeki olumsuz etkilerini en aza indirmek için stratejik yer seçimine odaklandık. Bu zorlu NP-zor problemlerini ele almak için genetik algoritma ve açgözlü algoritma olmak üzere iki algoritma keşfettik. Ayrıca bu algoritmalardan elde edilen çözümleri kesin yöntemlerle karşılaştırdık.

Genetik algoritma, dengeli çözümler için çeşitli kriterleri ve kısıtlamaları bir araya getirerek potansiyel tesis alanlarını sistematik olarak araştırır. Yinelemeli yaklaşımı, optimum veya optimuma yakın sonuçları bulmak için çözümleri geliştirir. Açgözlü algoritma ise, özellikle anlık kazançlar çok önemli olduğunda yerel olarak en uygun seçimleri verimli bir şekilde yapar. Kapsamlı deneyler yoluyla, her iki algoritmanın performansını kesin yöntemlerle karşılaştırdık. Kesin yöntemler optimallik garantisi sağlarken, büyük ölçekli problemler için hesaplama açısından yoğun olabilirler. Buna karşılık, genetik algoritma, çözüm kalitesi ile hesaplama çabası arasında umut verici bir denge sunarak geniş bir çözüm alanını verimli bir şekilde araştırır. Açgözlü algoritma, hesaplama açısından verimli senaryolarda tatmin edici sonuçlar sağlar.

Karar vericiler, her iki algoritmanın güçlü yanlarını birleştirerek ve performanslarını kesin yöntemlere göre değerlendirerek topluluklar ve çevre üzerindeki zararlı etkileri en aza indiren en uygun tesis konumlarını belirleyebilir. Bu araştırma, zararlı tesislerin neden olduğu zararı hafifletmek için etkili bir yaklaşıma katkıda bulunmaktadır.

Anahtar Kelimeler: Genetik, greedy, zararlı tesis

ABSTRACT

Obnoxious Facility Location Models and Solution Algorithms

YELİZ YILERİ TERKEŞLİ
M.Sc., Department of Indistrual Engineering

Supervisor: Asst. Prof. Dr. Nuşin UNCU

Co-Supervisor: Assoc. Prof. Dr. Fatih KILIÇ

August 2023, 59 pages

In this thesis we focused on the strategic selection of locations for obnoxious facilities to minimize their negative impact on communities and the environment. We explored two algorithms, the genetic algorithm and the greedy algorithm, to address these challenging NP-hard problems. Additionally, we compared the solutions obtained from these algorithms with exact methods.

The genetic algorithm systematically explores potential facility sites, incorporating diverse criteria and constraints for balanced solutions. Its iterative approach refines solutions to find optimal or near-optimal outcomes. The greedy algorithm, on the other hand, makes locally optimal choices efficiently, especially when immediate gains are crucial.

Through comprehensive experimentation, we compared the performance of both algorithms with exact methods. While exact methods provide guarantees of optimality, they can be computationally intensive for large-scale problems. In contrast, the genetic algorithm efficiently explores a wide solution space, offering a promising balance between solution quality and computational effort. The greedy algorithm provides satisfactory results in computationally efficient scenarios.

By combining the strengths of both algorithms and evaluating their performance against exact methods, decision-makers can identify optimal facility locations that minimize harmful effects on communities and the environment. This research contributes to an effective approach for mitigating harm caused by obnoxious facilities.

Keywords: Genetic Algorithm, greedy algorithm, obnoxious facilities

Dedication

Dedicated to my grandmother Dursun ŞEN

ACKNOWLEDGEMENTS

I would like to express my gratitude to my esteemed supervisor, Asst. Prof. Dr. Nuşin UNCU, who shared valuable knowledge with me throughout this study and patiently and attentively provided me with assistance whenever I needed it. Her guidance and contributions to my academic life, motivation, continuous encouragement, trust, and patience are highly appreciated.

I would like to thank my co-advisor, Assoc. Prof. Dr. Fatih KILIÇ to give his precious knowledge and to guide me at the preparation of the thesis.

I would also like to thank, Asst. Prof. Dr. Gizem Gül KOÇ and. Assoc. Prof. Dr. Melik KOYUNCU for accepting to be the member of examining committee for my thesis.

I would like to thank my family, for their love, support, and understanding throughout the years, special thanks to my mother Şeniz ŞEN who has been supporting me in all challenges in my life.

I would like to express my heartfelt gratitude to my spouse, Oğuzhan Terkeşli for his constant encouragement, understanding and support. His unwavering belief in me and my abilities has been a source of inspiration and motivation.

Last but not least, I would like to thank my four legged friends Boncuk and Çakıl for their limitless support and love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACO	: Ant Colony Optimization
DP	: Dispersion Problem
DSS	: Decision Support System
FDP	: Facility Dispersion Problems
GA	: Genetic Algorithm
MIP	: Mixed Integer Programming
MST	: Minimum Spanning Tree
PSO	: Particle Swarm Optimization
QAP	: Quadratic Assignment Problem
SA	: Simulated Annealing
TS	: Tabu Search

LIST OF SYMBOLS

W	**:** Scaling window
S	**:** Selection strategy
S	**:** Negative impact zone
p	**:** Sites
N	: Population size
M	**:** Mutation rate
i, j	**:** Indexes
G	**:** Generation gap
C	**:** Crossover rate

1. INTRODUCTION

The obnoxious facility location problem is a significant issue in various domains, including urban planning, waste management, and facility locating. This problem involves determining the optimal location of facilities while considering the negative impact they may have on the surrounding environment or population. Obnoxious facilities can include waste treatment plants, industrial facilities, or other operations that emit pollutants, generate noise, or produce other undesirable effects.

The location of obnoxious facilities is a complex optimization problem that requires careful consideration of environmental, social, and economic factors. Finding an optimal solution involves balancing the need for such facilities with minimizing their negative effect on the surrounding communities and ecosystems.

The motivation behind this research is to develop effective and efficient solution algorithms for the obnoxious facility location problem. The existing literature suggests various approaches, including mathematical programming, heuristics, and metaheuristics. However, further investigation is needed to explore the strengths and weaknesses of different algorithms in this context.

In this study aimed to contribute to the existing knowledge by comparing and evaluating the performance of two solution algorithms: the greedy algorithm and the genetic algorithm. The greedy algorithm offers a simple and intuitive approach, while the genetic algorithm leverages the evolutionary concepts of selection, crossover, and mutation.

By analyzing and comparing the performance of these algorithms on synthetic instances and real-world case studies, this research aims to provide insights into their effectiveness, efficiency, and suitability for addressing obnoxious facility location problems. The findings of this study can guide decision-makers, urban planners, and policymakers in making informed decisions regarding the optimal location of obnoxious facilities while minimizing their adverse impacts.

The subsequent sections of this thesis will delve into the literature review, methodology, experimental results, case studies, and conclude with a summary of contributions and future research directions.

Location problems frequently involve the consideration of undesirable or obnoxious facilities within networks. These facilities are labeled as such because they have the potential to pose risks or hazards to individuals residing in close proximity, negatively impact property values, or lead to a decrease in the overall quality of life through pollution. Examples of obnoxious facilities include nuclear reactors, garbage, prisons, dumps installations.

The presence of these facilities can generate concerns among communities due to the potential adverse effects they may have on the environment and the well-being of nearby residents. When determining the optimal location for these facilities, decision-makers face the challenge of striking a balance between the necessity of such infrastructure and minimizing the negative impacts on the surrounding population.

Location problems involving obnoxious facilities require careful consideration of factors such as proximity to residential areas, potential health and safety risks, and the potential for environmental contamination. Decision-makers must weigh the benefits provided by these facilities against the potential negative consequences to ensure that the chosen location minimizes the impact on individuals, property values, and overall quality of life in the surrounding area.

In various applications, demand points rely on facilities to receive essential services. However, it is worth noting that these facilities can also be considered "obnoxious" as they may generate nuisances that affect one or multiple demand points.

While facilities play a crucial role in meeting the needs of the surrounding areas, they can simultaneously create disturbances or inconveniences. These obnoxious effects can manifest in different forms, such as noise pollution, visual impact, traffic congestion, or environmental pollution. Examples of facilities that may be obnoxious in this context include power plants, waste treatment facilities, transportation hubs, or industrial complexes.

The presence of obnoxious facilities introduces a challenge in the optimization of location and service allocation decisions. Balancing the efficient provision of services with minimizing the adverse effects on demand points becomes a key consideration. Decision-makers must carefully assess and mitigate the potential nuisances caused by these facilities to ensure a harmonious coexistence with the surrounding communities.

Addressing the obnoxiousness of facilities requires incorporating various factors into the decision-making process. This includes evaluating the proximity of facilities to sensitive areas, implementing mitigation measures to minimize negative impacts, and actively engaging with the affected stakeholders to understand and address their concerns. By considering both the service demands of the demand points and the obnoxious effects of the facilities, decision-makers can strive for solutions that optimize service provision while minimizing nuisances for the affected communities.

2. LITERATURE REVİEW

Facility dispersion is a crucial concept in many fields, including military defense, franchise location, hazardous material transportation, and telecommunication network design. It involves strategically distributing facilities to maximize the separation between them, minimize potential interactions, and improve the reliability of service. Dispersion of facilities can be a good way to reduce negative impacts, promote safety, and ensure that critical facilities remain operational even under adverse conditions. The goal is to find the optimal distribution of facilities to meet the specific needs of a particular system or network. This technique has been widely studied and applied in different fields, and its effectiveness has been demonstrated in various contexts. In this sense, facility dispersion represents a valuable tool for improving the performance and efficiency of complex systems and networks.

The earliest study in the literature on location models for undesirable facilities is the discussion about "partially noxious" facilities introduced by Goldman and Dearing (1971). Their work laid the foundation for understanding the challenges and complexities of locating facilities that may have adverse effects on nearby communities and environments. Church and Garfinkel (1978) had presented the first solution achieved in $O(n \log n)$ time, utilizing novel material usage and methods.

McGinnis and White (1978) introduced a problem with two objectives, utilizing the minisum-minimax criteria. They define facility location in a planar region relative to multiple demand points. The minisum criterion seeks to find the minimum total distance between the facility and the demand points, while the minimax criterion aims to minimize the maximum distance between the facility and any of the demand points. By combining these two criteria, they presented a multi-objective optimization problem. Drezner and Wesolowsky (1983) presented

the rectilinear maximin problem as an approach to locate an obnoxious facility. The problem involves finding an optimal location for the facility in a two-dimensional plane, considering rectilinear distances, in a way that maximizes the minimum distance between the facility and any given demand point or customer. Ting (1984) proposed an algorithm to solve the problem of maximizing the sum of distances between an anonymous facility and a set of demand points on tree networks. The problem aims to find the best location for a facility on a network, in order to maximize the sum of distances between the facility and a set of demand points or customers.

Mehrez et al. (1986) presented an improved algorithm that utilized bounds to reduce the model size locating an obnoxious facility. This enhanced algorithm showcased significant improvements over the previous approach Drezner and Wesolowsky (1983), providing a more efficient and effective solution. Batta and Chiu (1988) explored the problem of routing an undesirable vehicle on a network. The primary objective was to identify an optimal path that minimizes the weighted sum of distances over which the vehicle remains within a specific threshold distance from population centers. This approach aimed to find efficient routes for the undesirable vehicle while limiting its proximity to populated areas, thereby addressing concerns related to safety and minimizing potential negative impacts on the surrounding population. Ratick and White (1988) proposed a multiobjective model that aimed to address two objectives simultaneously. The first objective was to minimize costs associated with a certain problem, while the second objective was to minimize the repulsion or negative impact experienced by people or the public. Wyman and Kuby (1995) introduced a model that featured three distinct objectives. The first objective was to minimize risk, indicating a focus on reducing potential hazards or adverse consequences. The second objective was to minimize costs, aiming to achieve economic efficiency. Lastly, the model aimed to maximize equity, suggesting a concern for ensuring fairness and equal distribution of benefits or burdens among different stakeholders.

Gopalan et al. (1990) introduced a model to measure the equity of HazMat transport risk. They proposed a heuristic solution to achieve a fair distribution of risks among different areas. The model aims to minimize the total risk of travel while ensuring that the risk is evenly spread across the geographic region where the transportation network is embedded. The heuristic approach provides an efficient and effective way to optimize the routing of hazardous materials, ensuring a balanced distribution of risks for different regions.

Melachrinoudis and Smith (1995) extended an algorithm for the obnoxious facilities location problem to include "m" edges and "n" nodes, assuming Euclidean spatial distances. However, research on this problem did not stop there, and various types of these problems with different assumptions were reviewed in the subsequent years.

Marianov and ReVelle (1998) proposed a linear vehicle routing problem model that minimized both cost and risk, utilizing advanced material usage and methods. Erkut and Verter (1998) extensively studied transport risk and reviewed various methods for its modeling. Rogers (1998) introduced a model focused on minimizing the global repulsion of inhabitants in a specific region. The objective was to tackle negative reactions stemming from the region's insecurity. Through the model, the study identified factors contributing to insecurity and aimed to reduce them, which subsequently led to a decrease in overall repulsion among residents. This approach had the potential to enhance the perceived safety and well-being of the area's inhabitants. Giannikos (1998) developed the goal-programming technique to solve the obnoxious facility location-routing problem. This method enabled the simultaneous optimization of multiple objectives, providing a comprehensive solution to efficiently address the complexities of locating and routing obnoxious facilities. Melachrinoudis (1999) proposed a bicriteria location model that specifically targeted the location of a new semi-obnoxious facility within an existing layout. The model was designed to optimize two criteria simultaneously, striking a balance between minimizing transportation costs by locating the facility close to existing ones while also considering the undesirable effects of being too close. To solve this complex problem, an adapted Fourier-Motzkin elimination method was employed, incorporating advanced techniques and approaches to achieve efficient and effective results. Zagrafos and Samara (2007) presented a methodological framework for creating a decision support system (DSS) for hazardous materials emergency response operations as noted by Erkut and Alp (2007).

Fernandez et al. (2000)’s focus was on the continuous location of an undesirable facility within a specific geographical region, considering environmental factors, material usage, and innovative methods. Akgün et al. (2000) conducted a review on cases where decision-makers sought various transportation routes in the same year. Zhang et al. (2000) developed an algorithm for routing hazardous materials using GIS technique and a Gaussian Plume model for their dispersion model in the context of transporting urban hazardous wastes, considering

novel material usage and methods. Katz et al. (2002) proposed a maximin model for multi-facility location, considering the distance between each facility and demand centers, as well as the distances among facilities, incorporating advanced material usage and methods.

Cappanera et al. (2004) focused on finding the best locations for obnoxious facilities and the optimal routes for transporting obnoxious materials between these facilities and various built-up areas, utilizing innovative material usage and methods. Rakas et al. (2004) developed a multiobjective model that used fuzzy data to identify the best locations for undesirable facilities, incorporating novel material usage and methods. Castillo (2004) examined problems related to transportation risks associated with hazardous materials and reviewed modeling tools available for the transportation of such materials, incorporating advanced material usage and methods. In addition, Castillo developed a route optimization model to assist decision making. Rodriguez et al. (2006) studied the problem of obnoxious facility location and considered Euclidean travel distances, alongside innovative material usage and methods.

Colebrook et al. (2005) introduced a novel approach for solving the network maxisum problem, which involved a new bound and an O(mn) algorithm, incorporating cutting-edge material usage and methods. Erkut and Ingolfsson (2005) developed an integer programming problem to address the tree design problem, aiming to balance cost and risk, incorporating advanced material usage and methods.

Díaz-Báñez et al. (2005) addressed the computation of shortest paths for transporting hazardous materials in continuous spaces. They proposed an approximate algorithm that employed the bisection method to transform the optimization problem into a decision problem. The objective was to compute the shortest path while ensuring that the minimum distance to the demand points was not smaller than a specified value r, taking into account safety concerns related to hazardous materials and utilizing advanced techniques and approaches.

Pisinger (2006) introduced an exact algorithm that provided a number of fast upper bounds for solving P-dispersion problems. These bounds could be computed in O(n) time, utilizing cutting-edge material usage and methods. Bell (2007) studied the risk-averse shipment of hazardous materials, considering mixed route strategies for repeated shipments on arcs with unknown incident probabilities. He suggested that the safest approach would be to use a combination of routes, incorporating advanced material usage and methods.

Berman et al. (2007) developed an optimized model for the transfer point location problem, which involves optimizing the location of a helicopter pad (transfer point) to efficiently serve a set of demand points requiring emergency service from a central facility, such as a hospital. They focused on finding the best location for one transfer point when the facility's location is known. Alumur and Kara (2007) proposed a model aimed to optimize the locations and technologies of treatment and disposal centers, as well as the routing of different waste types, with the objective of minimizing total cost and transportation risk. Akgün et al. (2007) focused on the effects of weather systems on HazMat transportation and introduced circle areas with different probabilities of variables to model such effects, utilizing cutting-edge material usage and methods.

Carotenuto et al. (2007) directed their attention to the problem of hazmat shipment routing and scheduling. They developed a tabu search algorithm to optimize the assignment of routes for hazmat shipments and create efficient schedules for their transportation. The study utilized innovative methods and techniques to tackle this intricate issue.

3. MATERIALS AND METHODS

3.1. Facility Dispersion Problem

The issue of facility location with hazardous materials involves the thoughtful location of specific substances, with a preference for keeping them far away from populated regions. This measure aims to protect the well-being of residents by reducing health-related concerns associated with these materials. The list of such substances includes harmful facilities and waste materials. Placing these materials in proximity to populated areas can lead to significant risks for human life (Rana & Garg ,2014).

In recent times, there has been a growing focus on a specific type of waste material known as hazardous or dangerous materials, commonly referred to as obnoxious. This category includes materials such as explosives, flammable liquids and solids, oxidizing agents, radioactive substances, corrosive materials, and poisonous and infectious substances in general.

In the operation of various facilities, transportation costs to and from the site are a major concern. However, in the case of a facility that may be considered unpleasant, its undesirability becomes a more critical factor. For instance, when selecting a site for a waste management

facility, many individuals would oppose a location based on minimizing transportation expenses. The perceived negative impact of living near such a facility far surpasses the benefits of reduced service costs. Consequently, these facilities are typically situated near the outskirts of the urban area (Erkut& Neuman, 1989).

In practical scenarios, every facility location solution must yield a more positive effect in terms of services compared to the negative effect of the costs it incurs. Considering the location of a facility that produces disadvantages than advantages would be unreasonable, if not it is a component of system that indeed generates overall advantages (Church& Drezner,2022).

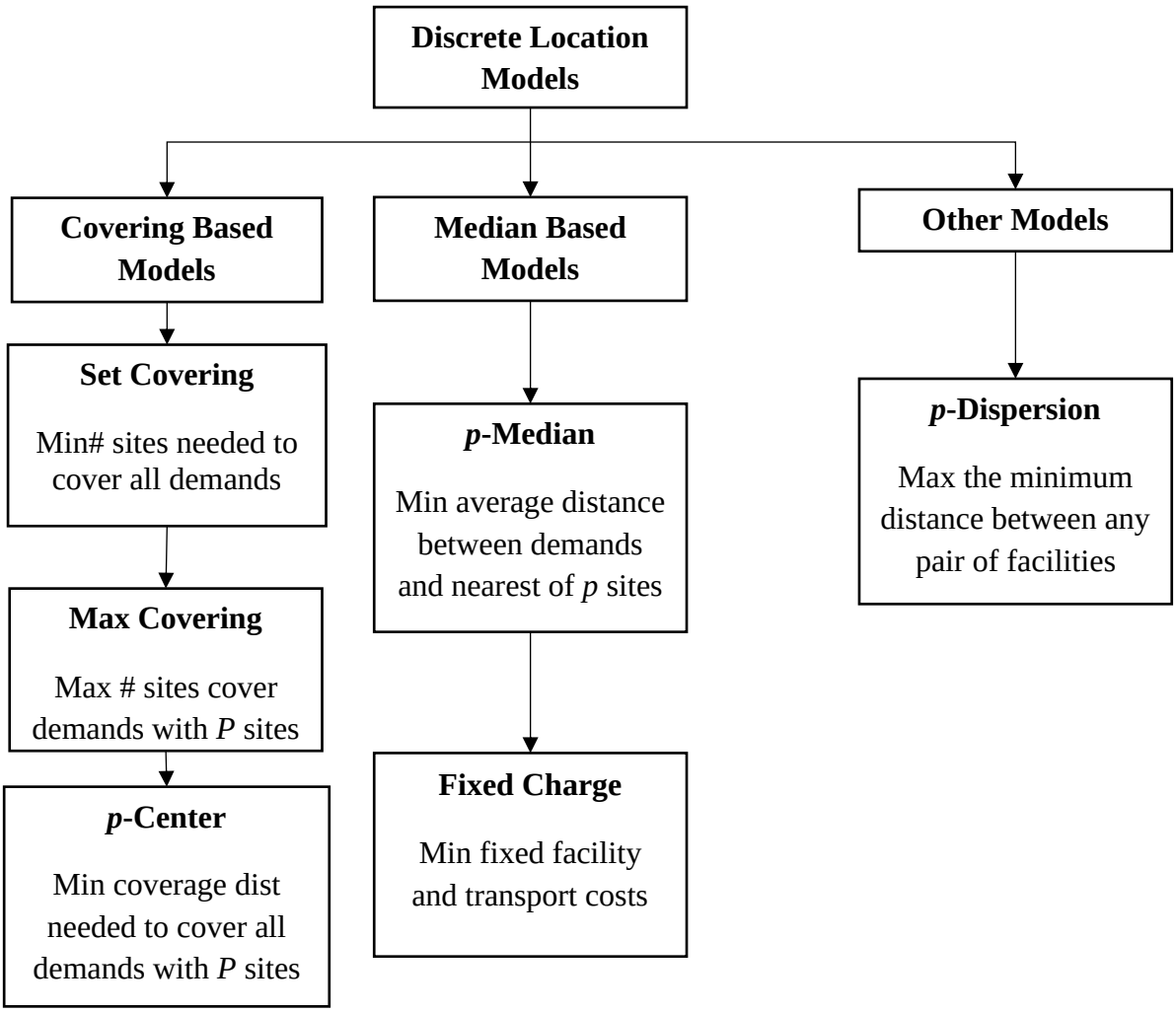

Figure 3.1. Discrete location models

Farahani & Hekmatfar explains the application types as follow:

- Identifying optimal locations for chemical plants

- Evaluating suitable sites for nuclear reactor plants
- Identifying efficient locations for pollution plants
- Locating appropriate dump sites
- Investigation of routing and location routing strategies for these applications has been on the agenda in recent years.

Farahani and Hekmatfar outlined that facilities can be broadly categorized into two groups based on their desirability to the nearby inhabitants. The first group comprises facilities that are preferred to be located in close proximity, such as those related to healthcare, public safety, commercial services, and education. The second group includes facilities that are generally considered undesirable by the surrounding population which people tend to avoid.

Dispersion problems which presented in Figure 3.1. involve locating facilities in a way that minimizes their mutual impact. This category often includes cases such as sales representatives of organizations. The main goal is for the organization to minimize competition among its sales representatives and increase coverage by maximizing the distance between them. In certain cases, investors may intentionally try to minimize the distance between their sales representatives. For example, one can imagine two restaurants with the same ownership, placed in one area to attract more customers. Interestingly, this strategy not only retains existing customers but also generates additional profits.

Undesirable facilities problems arise due to several concerns, and the aim is to place such facilities far away from demand centers. Despite being necessary for the community, these undesirable facilities, like garbage dump sites, can lead to disagreements within the population regarding their location. Decision-makers face a bi-objective problem: minimizing transport cost and maximizing distance from demand centers. Additionally, ensuring security during the transportation of hazardous materials from these centers becomes a crucial aspect of transport routing (Farahani & Hekmatfar,2009).

Four basic models which listed in Table 3.1. have been developed in the literature to differentiate the facilities we focus on in this study.

The first model, involves locating p-facilities while maximizing the minimum distance between any two facilities, known as the p-dispersion problem.

The problem is a model developed in the literature to disperse facilities from each other. The problem involves locating p-facilities while maximizing the minimum distance separating any two facilities. In other words, the goal is to seek the optimal locations of p-facilities that will result in the maximum distance between any two facilities. This can help minimize the negative impact that facilities have on each other, such as reducing the likelihood of accidents or minimizing the potential for interaction among hazardous facilities. The p-dispersion problem is called the Max-Min-Min problem, as it aims to maximize the minimum distance between any two facilities. By applying the p-dispersion problem, it is possible to enhance the reliability of service and logistic systems and promote the robustness and reliability of critical facilities (Shier,1977).

The second model, focuses on defining the minimum separation distance for each facility, called the p-defense problem. The problem focuses on defining the minimum separation distance for each facility, with the goal of locating p-facilities to maximize the sum of these minimum separation distances. In other words, the objective is to find the optimal locations for p-facilities that will maximize the total minimum separation distances between all facilities. The p-defense problem has also been named as the Max-Sum-Min problem, as it describes maximizing the sum of minimum separation distances. Overall, the p-defense problem provides a useful tool for facility location planning that can help optimize resource allocation, minimize risk, and improve overall performance (Moon and Chaudhry,1984).

The third model, involves maximizing the sum of all separation distances, known as the p-dispersion sum problem. It involves locating p-facilities in a way that maximizes the sum of all separation distances. The objective is to find optimal locations for p-facilities that will maximize the total sum of separation distances for all pairs of facilities. This problem has also been classified as the Max-Sum-Sum problem, as it maximizes the sum of sums of all separation distances.

The p-dispersion sum problem is useful in minimizing the potential interaction among hazardous facilities, enhancing the reliability of service and logistic systems, and promoting the robustness and reliability of critical facilities. By maximizing the sum of all separation distances, the p-dispersion sum problem can help minimize the negative impact of facilities on each other and improve overall performance in facility location planning (Kuby,1987).

The fourth model deals with locating p-facilities while maximizing the smallest of the facility-defined sums. The facility-defined sum represents the sum of separation distances from a specific facility location to all other facilities. The objective of this problem is to locate p-facilities optimally that will maximize the smallest facility-defined sum. The Max-Min-Sum problem has also been classified as a combinatorial optimization problem. This problem is useful in minimizing the negative impact of facilities on each other, enhancing the reliability of service and logistic systems, and promoting the robustness and reliability of critical facilities. By maximizing the smallest facility-defined sum, this problem can help to improve overall performance in facility location planning by ensuring that the most important facilities are located in the optimal positions (Erkut and Neuman,1991).

Erkut and Neuman (1991) developed a scheme, presented in Table 3.1, that helps to clarify the structure of location models used for dispersing facilities. All of these models involve maximizing two elements. The first element determines whether the problem considers all facilities or just one facility, while the second item specifies if it is based on the sum of split distances for a facility or the minimum separation distance a facility experiences. For example, the Max-Min-Min problem, involves maximizing the minimum distance across all facilities, where each value represents the distance to the closest facility. However, the Max-Sum-Min problem seeks to maximize the sum of the separation distances to the closest facility for each facility.

Curtin and Church (2006) introduced a new approach to facility dispersal by considering a multi-type dispersion metric, which recognizes that the optimal distance between two facilities should depend on their respective types. In this approach, the intensity of interaction between facilities is represented by incorporating both the conventional separation distance and a type-specific "repulsion" factor, where a smaller repulsion value signifies a stronger inter-facility interaction.

Curtin and Church (2006) developed four dispersion models based on the four fundamental dispersion metrics classified by Erkut and Neuman (1991), as well as a multi-type metric where each model is an extension of a classic dispersion model that takes into account different types of facilities.

Model name	Technical description	Classification	Reference
p-dispersion	Maximize the minimum separation distance	Max-min-min	Shier 1977; Moon and Chaudhry 1984
p-defense	Maximize the minimum separation distances (one for each facility)	Max-sum-min	Moon and Chaudhry 1984
No name has been given	Maximize the minimum facility sum	Max-min-sum	Erkut and Neuman 1991
p-dispersion sum	Maximize the sum of all separation distances	Max-sum-sum	Kuby 1987

Table 3.1. Facility dispersal models: The Single Type of Facility Case

However, there may be some limitations to the existing dispersion models. In the next section, the authors will identify these potential shortcomings and propose a new measure called the partial sum dispersal measure. They will show how this new measure can be used to develop more general facility dispersal models that bring together seemingly disparate research in this field.

3.1.1. P-dispersion Problem

Kuby (1987) addressed a facility location problem on a network, aiming to maximize the minimum separation distance between open facilities. This is relevant in scenarios involving facilities posing threats to each other or in systems of retail or service franchises, where the goal is to place facilities as far apart as possible from their nearest counterparts. A mixed-integer program is formulated to achieve this objective, with 0-1 location variables reversed in distance constraints to ensure only open facility pairs limit the maximization. Additionally, Kuby introduced the "maxisum dispersion problem" to maximize average separation distance between open facilities. Computational results evaluate scenarios with 5 and 10 facilities on a 25-node network. A multicriteria approach combining dispersion and maxisum problems is

explored. The p-dispersion problem has a weak duality with the (p-1)-center problem, offering potential for finding initial distances in covering problems for the p-center problem.

3.1.2. Maximum Min-Sum Dispersion Problem

The max-min-sum diversity problem (DP) involves identifying a subset S from a set N, where the minimum sum of distances to the other selected elements is maximized.

In a given set N comprising *n* elements labeled with integer numbers from *1* to *n*, and with defined distances d*ij* between any two elements *i* and *j*, a diversity problem (DP) involves identifying a subset S ⊂ N. The main goal of the DP is to maximize or minimize an objective function based on the distances between elements within S. Based on the specific objective function utilized, dispersion problems are categorized into two primary classes: efficiency-based and equity-based.

In the case of efficiency-based objective functions, the goal is to maximize the dispersion quantity for the entire selected subset S. On the other hand, equity-based objective functions aim to ensure equitable dispersion among the elements chosen in the subset. (Amirgaliyeva et al., 2017)

3.1.3. P-defense Dispersion Problem

Moon and Chaudhry (1984) introduced the concept of the p-dispersion problem, which aims to disperse facilities from each other on a network. The problem involves locating a certain number of facilities, denoted by "*p*", on the network such that the closest pair of facilities has maximum separation distance. Additionally, they proposed a model for the p-defense problem, which involves maximizing the sum of minimum separation distances between the facilities while locating "*p*" facilities. For this problem, a separation distance is defined for each facility.

3.1.4. Maxisum Dispersion Problem

The maxisum dispersion problem is analogous to the p-dispersion problem (maximin), similar to how the p-median problem (minisum) corresponds to the p-center problem (minimax). In the discrete maxisum problem, the goal is to place p facilities among n discrete nodes to maximize the sum of distances (or average of distances) between the open facilities. This maxisum

objective function offers an alternative criterion for achieving greater dispersion among the facilities.

However, a potential drawback of this interpretation of "spreading apart" is that certain pairs of facilities might still end up being placed very close to each other despite the overall dispersion objective (Kuby,1987).

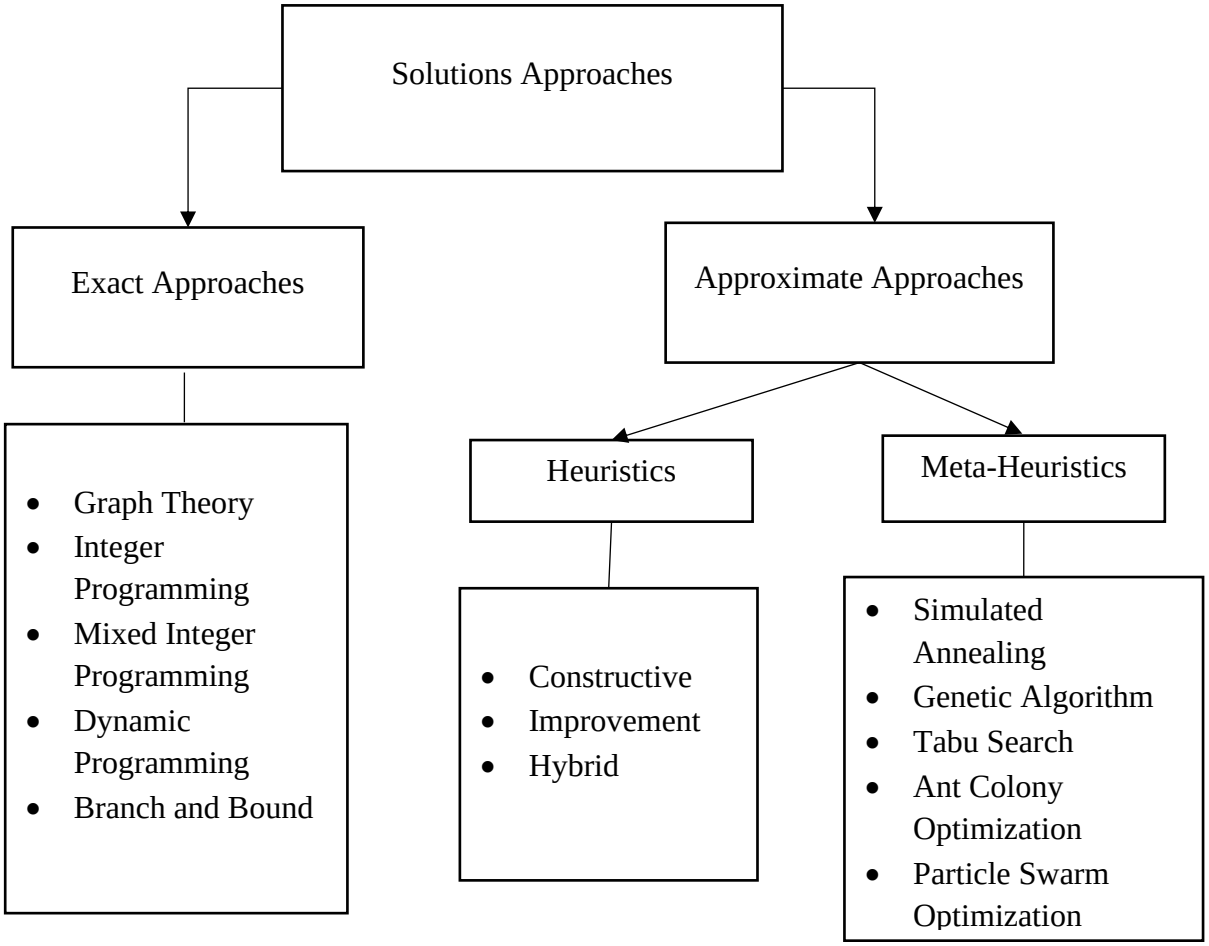

Figure 3.2. Solution approaches

Over the past decades, numerous specialized optimization methods have been developed and implemented to address the diverse range of Facility Dispersion Problems (FDPs). Optimization techniques like genetic algorithm (GA), simulated annealing (SA), ant colony optimization (ACO), tabu search (TS), and particle swarm optimization (PSO) are commonly employed metaheuristics in this context. Furthermore, the combination of multiple approaches through hybrid heuristics and metaheuristics has exhibited impressive performance and offers promising prospects for effectively solving complex FDPs in the future. The solution approaches for FDPs are shown in Figure 3.2.

3.2. First Model: The Minimum Impact Location Problem

Church and Cohon (1976), in their study, highlighted the significant interest in public facility location modeling in recent years. They emphasized that the effectiveness constraint used is the main differentiating factor among various approaches to this modeling. This constraint assesses the efficiency of a particular location configuration in relation to the overall purpose of the service and the intended coverage area. Various constraint has been developed for selection depending on the type of service provided.

One commonly used effectiveness constraint is the total weighted distance or time required to travel to the facilities. This constraint quantifies the general accessibility of the facilities, with smaller values indicating higher accessibility. Another constraint is the maximal service distance, which represents the distance or time that the farthest user would have to travel to reach a facility. It reflects the worst possible performance of the system given a particular location configuration.

Many location problems incorporating the concept of maximal service distance can be categorized as "covering problems." One such problem is the location-set covering problem, introduced by Toregas et al. (1972). This problem aims to identify the minimum number and optimal locations for facilities to ensure that no demand point is located beyond the maximal service distance from a facility. Case and White (1974) referred to this as the total cover problem.

Recognizing that it is often impossible to provide enough facilities to fully cover all demand within the desired maximal service distance, Church and ReVelle (1974) introduced the maximal covering location problem. This problem involves maximizing the coverage (population covered) within a specified distance by strategically locating a fixed number of facilities. They presented a linear-integer programming formulation for this problem, applicable to both network and Euclidean plane scenarios The objective of this problem is to identify and locate a certain number of facilities to satisfy the highest possible number of demand points. They proposed to use linear programming and branch-and-bound algorithms to determine optimal solutions and also used various heuristics to find good or near-optimal solutions.

Case and White (1974) studied a closely related problem known as the partial coverage problem. The objective of this problem is to identify and locate a certain number of facilities to

cover the maximum possible number of demand points. The partial covering problem is considered a specific instance of the maximal covering problem, where each demand point is assigned equal covering weights. They also investigated heuristic methods to address the partial covering problem.

Both the maximal covering approach by Church and ReVelle (1974) and the partial covering approach by Case and White (1974) assume a finite number of predefined potential facility sites.

Church and Cohon (1976) developed a comprehensive energy-production model with multiple objectives. Their model aims to find suitable locations for power plants, simulate heat discharge to water sources, limit air pollutant emissions in downwind areas, minimizing the costs associated with inter-basin water transfers, reduce expenses related to power transmission, and minimize the costs of constructing power plants. Furthermore, their model focuses on minimizing the negative impact on societies while meeting their energy demand. In exploring strategies to minimize the negative impacts of a nuclear power plant, they introduced a new approach called the minimum effective location problem (MILP).

This model involves determining the location of *p* facilities from a set of *m* potential sites in a manner that minimizes the number of individuals within a certain distance of one or more of these facilities. To facilitate further discussion, the following notation is considered:

S: a negative impact zone around a facility

I: index number of a community at a given location, $i = 1,2,3,\ldots, n$

J: index number of a candidate obnoxious facility site, $j = 1,2,3,\ldots,m$

a_i: the population size at i^{th} community

d_{ij}: the distance between the center of community *i* and obnoxious facility site *j*

$N_i = \{j | d_{ij} \leq S\}$, the set of sites that are closer than desired to community *i to j*

$$\bar{x_j} = \begin{cases} 1 & if\ site\ j\ is\ not\ chosen\ for\ a\ new\ facility\ and\ left\ vacant \\ 0 & if\ site\ j\ is\ chosen\ for\ a\ facility \end{cases}$$

$$\bar{y_j} = \begin{cases} 1 & if\ community\ i\ is\ covered \\ 0 & if\ community\ i\ is\ \ not\ covered \end{cases}$$

Formulation of the MILP is as follows:

$$Minimize\ Z = \sum_{i=1}^{n} a_i \bar{y}_i \quad (1)$$

Subject to: (2)

$\sum_{j=1}^{m} \bar{x}_j = m - \text{p}$

$\bar{x}_j + \bar{y}_i \geq 1 \qquad \forall\ j \in N_i \quad (3)$

$\bar{x}_j \in \{0,1\} \qquad \forall j == 1, 2, \ldots, m \quad (4)$

$\bar{y}_i \in \{0,1\} \qquad \forall i == 1, 2, \ldots, n \quad (5)$

The objective of the minimum-impact location problem is to minimize the population size within a distance *S* from any facility, while locating a fixed number of facilities. It should be noted that this formulation includes decision variables representing sites not used for facility configuration. Equation (2) provides that all but *p* sites remain vacant, and each site can be selected for only one facility. Church and Cohon (1976) opted to structure this problem in a way that most of the constraints resemble set covering conditions, which facilitates the identification of integer solutions.

In their Brookhaven report, addressed several other objectives pertaining to the location of obnoxious facilities such as nuclear and coal-fired power plants. These objectives are as follows:

1. Population exposure: They suggested minimizing the population size within a given distance from nuclear power plants. This objective aimed to reduce the population at risk of significant radiological accidents. A similar model called the expropriation problem was later introduced, which involved budgeting for the expropriation of properties near planned obnoxious facilities to maximize the distance between the facility and the nearest non-expropriated property. Another proposed form involved minimizing the exposed population in a region multiplied by the number of plants within the exposure distance. This form takes into account the assumed relationship between risk and the number of nearby generating units. In an application to the six mid-Atlantic states, the third proposed form, which minimized the weighted population within the exposure distance, was adopted.

2.Distance: This objective had served as a surrogate approach to maximize safety. The goal was to maximize the average distance between communities and the nearest nuclear power plants. While the logical expectation is to locate power plants in the most remote parts of a

region, practical constraints such as water availability, thermal load discharge limitations and site capacity may prevent this. As this is a multi-objective model, the goal was to find Pareto optimal solutions that balance cost and safety considerations.

3. Capacity: They emphasized the importance of avoiding the concentration of generating capacity in specific geographic or political areas. They noted that an imbalance in power plant siting could lead to criticisms related to equity, such as communities questioning why they should bear the negative effects of a nearby facility while others are unaffected. To address these concerns, they suggested incorporating regional limits on obnoxious facilities or ensuring that each site provides at least a given fraction of its power demands. By considering such equitable distribution, opposition to the facility siting can be minimized.

These additional objectives aimed to address various aspects of population safety, equity, and regional distribution when locating obnoxious facilities like nuclear power plants.

3.3. Solution Algorithms

3.3.1. Exact Algorithms

Exact methods, also known as exact algorithms or optimization methods, are a category of computational techniques used to find the optimal solution to a given problem within a finite amount of time. These methods guarantee that the solution found is the best possible solution according to the problem's objective function and constraints. Exact methods are particularly useful for solving optimization problems with a relatively small problem size, where it is feasible to explore all possible solutions.

There are various exact methods, each tailored to different types of optimization problems. Some of the common exact methods are:

Graph theory is a mathematical modeling that deals with the study of graphs consisting of nodes (vertices) and edges connecting these nodes. In the context of optimization, graphs can be used to model and solve problems such as the Traveling Salesman Problem (TSP) or the Minimum Spanning Tree (MST) problem. While the TSP aims to find the shortest possible route that visits each node exactly once and returns to the starting node, the MST problem looks for the shortest possible set of edges that connect all nodes in a graph with minimum total weight.

Integer programming is a type of mathematical optimization technique used to solve problems where some or all of the decision variables are required to take integer values. Unlike linear programming, where variables can take any real value, integer programming is particularly useful when dealing with discrete decision variables. It finds application in various real-world scenarios, such as production planning, resource allocation, and project scheduling, where decisions need to be made in whole units.

Mixed Integer Programming (MIP) is an extension of integer programming that allows a combination of integer and continuous decision variables. This powerful optimization technique can be applied to a wide range of complex problems involving both discrete and continuous choices. MIP finds use in a variety of fields, including logistics, supply chain management and network design, where decisions can involve both discrete choices (e.g., selecting locations) and continuous choices (e.g., allocating resources).

Dynamic programming is a method for solving optimization problems that exhibit optimal substructure and overlapping subproblems. It divides complex problems into smaller subproblems and saves the solutions of these subproblems to avoid unnecessary computations. Dynamic programming is particularly effective for solving sequential decision-making problems, such as resource allocation over time or finding the shortest path in a graph. Common applications of dynamic programming include the knapsack problem, shortest path algorithms, and resource management in dynamic environments.

The branch and bound algorithm is a method for solving optimization problems by systematically sorting and evaluating potential solutions. It involves building a dynamic tree structure where the root nodes represent the initial search space and each node represents a potential solution or a subset of the solution space.

During the execution of the algorithm, the search tree is pruned or trimmed based on the constraint function or objective function of the optimization problem. This pruning process eliminates subtrees that are guaranteed to not contain optimal solutions, thereby reducing the search space and improving efficiency.

The branch and bound algorithm can be combined with other techniques, such as the polyhedral method, to form the Branch Cut method. This combined approach has been successfully applied to solve large-scale and challenging optimization problems.

In the context of facility location problems, the branch-and-bound algorithm has been used, for example, in the formulation of quadratic set overlap for facility arrangement. This formulation can handle various layout scenarios, including single-ply or multi-ply layouts, irregularly shaped sections, and modifications to existing layouts. The final layout solution is obtained through the execution of the branch-and-bound algorithm, which explores the solution space and identifies the optimal arrangement based on the defined constraints and objectives.

3.3.2. Greedy Algorithms

Greedy algorithms are known for their simplicity and efficiency, as they do not backtrack on earlier decisions while making a sequence of choices. They are widely used in various areas such as data compression, DNA sequencing, minimum-cost spanning trees, computational geometry, and network routing, among others. Greedy algorithms can also be used as heuristics to solve complex problems approximately, providing near-optimal solutions and in some cases even producing exact solutions.

An optimization problem is characterized by its potential solutions and a global optimality criterion that determines the optimal solution. A greedy algorithm makes a series of choices based on a local optimality criterion, determining the best choice at each step. These choices are so-called structural steps, steps to create a solution to the problem. The structural step aims to generate all potential solutions by applying to the input in all possible ways. A complete solution is a fully structured potential solution that may or may not be optimal according to the global optimality criterion, while a partial solution is a partially structured potential solution. Existing theories related to greedy algorithms have often focused on the following aspects:

• Presenting the formulation of greedy algorithms.

• Providing proofs to demonstrate the correctness of greedy algorithms.

• Describing and defining the characteristics of data structures suitable for greedy algorithms.

• Developing strategies for synthesizing new greedy algorithms.

• Ensuring comprehensive coverage of as many different types of greedy algorithms as feasible (Curtis,2003).

The greedy algorithm is a simple and heuristic approach to problem solving that follows a "greedy" strategy of making locally optimal choices at each step in the hope of finding a global optimum. It starts with an empty solution and incrementally builds the solution by selecting the best available option at each decision point.

The steps involved in the greedy algorithm are as follows:

Step 1 Initialization:

Start with an empty solution or a partially structured solution, depending on the problem.

Set any necessary initial parameters or variables.

Step 2 Selection:

Identify the available options or choices at the current step.

Evaluate each option on the basis of a specific constraint or heuristic that guides the decision-making process.

Select the option that is considered the best or most appropriate according to the constraint.

Step 3 Feasibility Check:

Check if the selected option satisfies any constraints or requirements imposed by the problem.

Make sure that the selected option is applicable and does not violate any predefined conditions or limitations.

If the option is feasible, proceed to the next step. Otherwise, backtrack or make adjustments.

Step 4 Update the Solution:

Incorporate the selected option into the solution or partial solution.

Modify the current solution based on the choice made in the previous step.

Update any relevant variables or data structures associated with the solution.

Repeat Steps 2-4:

Continue the process iteratively by repeating steps 2 to 4.

Select the best option at each step, considering the current state of the solution.

Perform the feasibility check to ensure the chosen option is valid.

Update the solution by incorporating the selected option.

Repeat these steps until a complete solution is obtained or a termination condition is met.

3.3.3. Genetic Algorithm

The theory of biological evolution proposed by Darwin has inspired the GA algorithm, which simulates the survival of the fittest. This mechanism is based on the fact that in nature, organisms that are fitter have a greater chance of survival and are more likely to pass on their genes to the next generation. As a result, good genes that help species better adapt to their environment become dominant in subsequent generations. To represent optimization problems, the GA algorithm uses chromosomes and genes, similar to those found in nature. Each solution corresponds to a chromosome and each gene represents a variable in the problem. To improve the chromosomes in each generation, the GA algorithm employs three main operators: selection, crossover, and mutation. These steps, along with problem representation and initial population, are discussed in the following sub-sections.

Metaheuristics possess several fundamental characteristics that can be summarized as follows:

• Metaheuristics are not customized to a specific problem; they are general problem-solving approaches.

• Typically, metaheuristics generate approximate solutions rather than exact solutions.

• Metaheuristics explore the search space to find solutions that are considered "good enough".

• Metaheuristics can be defined at an abstract level, which provides flexibility in their application.

• Parallel implementation of metaheuristics is usually simple and efficient.

• Metaheuristics cover a wide range of techniques, from basic local search methods to advanced learning approaches.

• To prevent early convergence, metaheuristics may involve various mechanisms.

• A metaheuristic can utilize heuristics as domain-specific knowledge, but the heuristics are guided by the higher-level strategy of the metaheuristic.

• Modern metaheuristics often use guide memory to improve performance, which preserves the search experience (Abdel-Basset, Abdel-Fatah & Sangaiah,2018).

Genetic algorithms are considered a search process used in computation to discover exact or approximate solutions for optimization and search problems. They are often referred to as global search heuristics. These techniques are inspired by evolutionary biology, including concepts such as inheritance, mutation, selection and crossover. By simulating these biological processes, genetic algorithms provide a means for programs to autonomously improve their parameters (Kumar, Husian, Upreti & Gupta, 2010).

Genetic algorithms (GAs) were initially introduced by a researcher named John Holland around 1960. These algorithms draw inspiration from nature and are based on the principles of Darwinian evolution. They typically consist of an initialization method and a fitness function, among other components. (Lamini, Benhlima, & Elbekri,2018)

The genetic algorithm is highly valuable for finding the shortest path by encoding paths in a graph and converting them into chromosomes. (Lambora, Gupta, & Chopra,2019)

A gene represents a specific unit of information that encodes a particular characteristic or trait of an individual within a population. Genes are typically represented as a string of binary digits or other discrete values, and they combine to form chromosomes, which represent the entire genetic makeup of an individual.

Genes are manipulated through a process of selection, crossover, and mutation in order to create new individuals with potentially improved fitness values. During selection, individuals with higher fitness values are more likely to be chosen for reproduction, and during crossover, the genes of two selected individuals are combined to form the genes of a new offspring. Mutation introduces random changes to the genes of an individual in order to explore the search space and potentially find new solutions.

The concept of genes is inspired by the principles of genetics and evolution in biology, and it is used to represent the key features and characteristics of individuals in a population. By

manipulating and evolving the genes of individuals over time, genetic algorithms can converge towards optimal solutions to complex problems.

A chromosome is a collection of genes that represents the complete genetic makeup of an individual in a population. Chromosomes are typically represented as strings of binary digits or other discrete values, and they encode the traits and characteristics of an individual that determine its fitness or suitability for a particular problem.

The structure and content of chromosomes are essential to the operation of genetic algorithms. The genes within a chromosome may be organized into specific loci or positions, and the sequence of genes may be important for determining the function and behavior of an individual. During the selection, crossover, and mutation operations of the genetic algorithm, chromosomes are manipulated in order to create new individuals with potentially improved fitness values.

The concept of chromosomes in genetic algorithms is inspired by the structure and function of chromosomes in biology. In biology, chromosomes are the structures that contain the genetic information of an organism, and they are passed on from one generation to the next during reproduction. Similarly, in genetic algorithms, chromosomes represent the genetic information that is passed on from one generation of individuals to the next, as the population evolves and converges towards optimal solutions.

Chromosomes are the key data structures used in genetic algorithms to represent the genetic makeup of individuals in a population. By manipulating and evolving the chromosomes of individuals over time, genetic algorithms can explore the search space and potentially find optimal solutions to complex problems.

A population is a collection of individuals that represent potential solutions to a given problem. The individuals within a population are typically encoded as chromosomes, which consist of a collection of genes that determine the traits and characteristics of the individual.

The population is a fundamental concept in genetic algorithms, as it is the pool of individuals from which new generations are created and evaluated. The initial population is generated randomly or through some heuristic method, and it represents the starting point for the evolutionary process.

During the genetic algorithm, individuals in the population are evaluated according to their fitness, a measure of how well they can solve a given problem. Individuals with higher fitness values have a greater probability of being selected for reproduction, while those with lower fitness values are more likely to be eliminated from the population. New individuals are created through the process of crossover and mutation, in which the genes of two or more selected individuals are combined to form the genes of a new offspring. The offspring are added to the population, and the process of selection, crossover, and mutation continues until a termination condition is met (e.g., a maximum number of generations has been reached, or a satisfactory solution has been found).

The concept of population in genetic algorithms is inspired by the principles of natural selection and evolution in biology. In biology, a population is a group of organisms of the same species that live in the same geographical area and interbreed. Similarly, in genetic algorithms, a population is a group of individuals that share the same problem domain and that can interbreed through the process of reproduction.

The initial population is the starting point for the evolutionary process. It is the initial collection of individuals that are randomly or heuristically generated to represent potential solutions to a given problem.

The size and composition of the initial population are important factors in the performance of the genetic algorithm. A larger population may lead to a more diverse and robust search space, but it may also increase the computational complexity of the algorithm. A smaller population may be more efficient, but it may also result in a narrower search space and a greater risk of premature convergence.

The individuals in the initial population are typically represented as chromosomes, which consist of a collection of genes that encode the traits and characteristics of the individual. The genes may be represented as binary digits or other discrete values, depending on the problem domain and the encoding scheme used.

The initial population may be generated randomly, by randomly generating chromosomes that meet some predefined criteria or by using some heuristic method to generate chromosomes that are likely to be good solutions. Alternatively, the initial population can be created based on

prior knowledge or experience with similar problems to guide the search towards specific regions of the search space.

The choice of the initial population and the encoding scheme used are important factors in the performance of the genetic algorithm. A good initial population can help the algorithm to quickly explore the search space and converge towards optimal solutions, while a poor initial population can lead to slow convergence or even early convergence.

A fitness function is a mathematical function that is used to evaluate the fitness or suitability of an individual within a population. The fitness function provides a quantitative measure of how well an individual solves the problem at hand, and it is used to guide the selection, crossover, and mutation operations of the genetic algorithm.

The fitness function typically takes the individual's chromosome as input and produces a single fitness value as output. The fitness value is a measure of the quality of the individual, and it is used to rank the individuals within the population. Individuals with higher fitness values are more likely to be selected for reproduction, while those with lower fitness values are more likely to be eliminated from the population.

The design of the fitness function is a critical aspect of the genetic algorithm, as it determines how well the algorithm is able to find optimal solutions to the problem at hand. The fitness function should be designed to capture the most important aspects of the problem and should be tailored to the specific problem domain. It may be necessary to experiment with different fitness functions in order to find the most effective one for a particular problem.

In some cases, the fitness function may be a simple mathematical formula that directly evaluates the individual's chromosome. In other cases, the fitness function may involve a more complex evaluation procedure that requires simulation or other computational methods.

The concept of the fitness function in genetic algorithms is inspired by the principles of natural selection and evolution in biology. In biology, fitness is a measure of an organism's ability to survive and reproduce in a given environment. Similarly, in genetic algorithms, fitness is a measure of an individual's ability to solve the problem at hand and to pass on its genetic material to future generations.

Genetic Algorithms (GAs) are search-based algorithms inspired by natural selection and heredity principles. GAs are a subset of the broader field of Evolutionary Computation. In GAs, a diverse set of solutions is generated for a given problem. These solutions undergo recombination and mutation processes, similar to biological genetics, producing new offspring, and the process is repeated for multiple generations. Each solution is assigned a unique fitness value based on its objective function, and individuals with higher fitness have a greater chance of mating to produce fitter offspring. This iterative process continues until a final destination criterion is met.

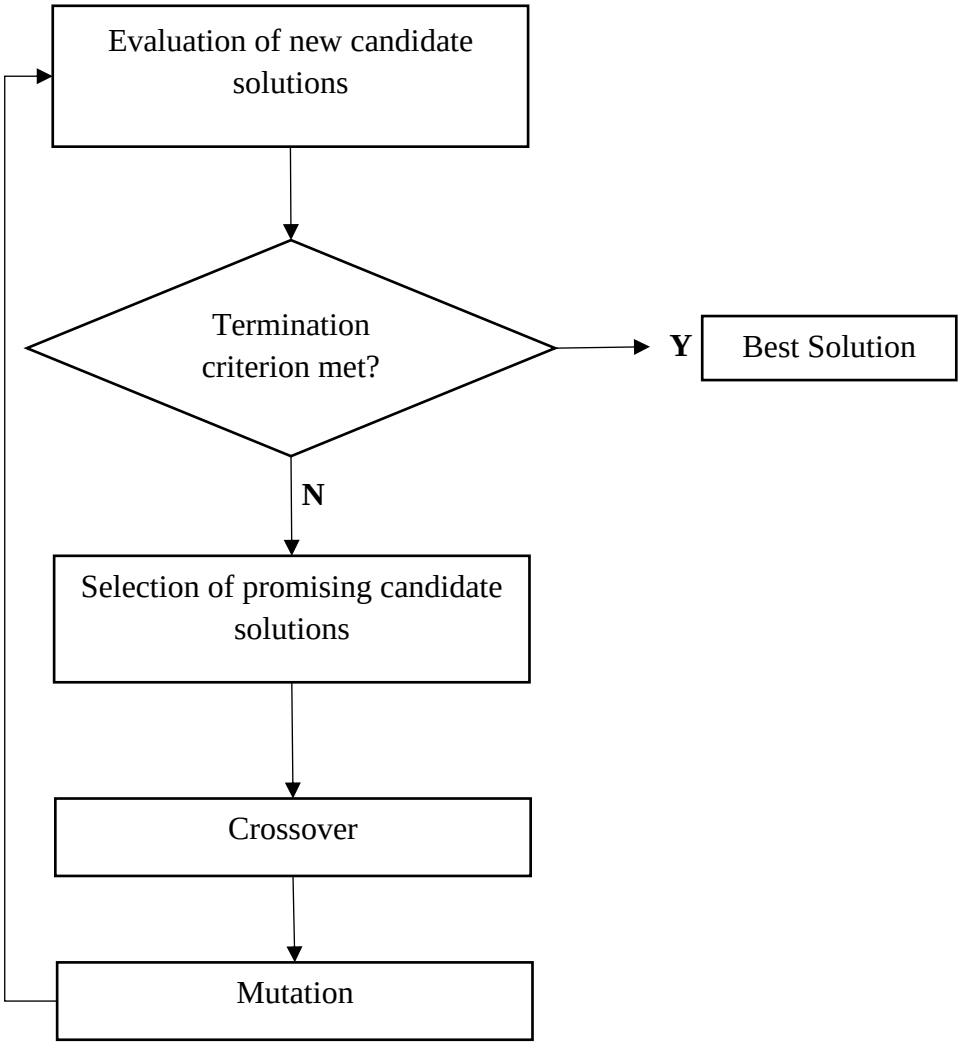

Figure 3.3. Genetic algorithm workflow

Genetic algorithms rely heavily on probability-based mechanisms which is presented a flow diagram in Figure 3.3., which distinguishes them from local random search methods that use

random solutions and may struggle to identify the best possible solutions. The incorporation of historical data in GAs makes them more complex and sophisticated, contributing to their effectiveness in exploring search spaces and finding better solutions.

Encoding solutions into chromosomes is a crucial aspect of genetic algorithms, and maintaining non-string coding approaches is a key concern. Three critical issues arise concerning the encoding and decoding process between chromosomes and solutions (or the mapping between phenotype and genotype):

• Ensuring the feasibility of a chromosome.

• Verifying the legality of a chromosome.

•Ensuring the uniqueness of mapping between chromosomes and solutions which modelled in Figure 3.4.

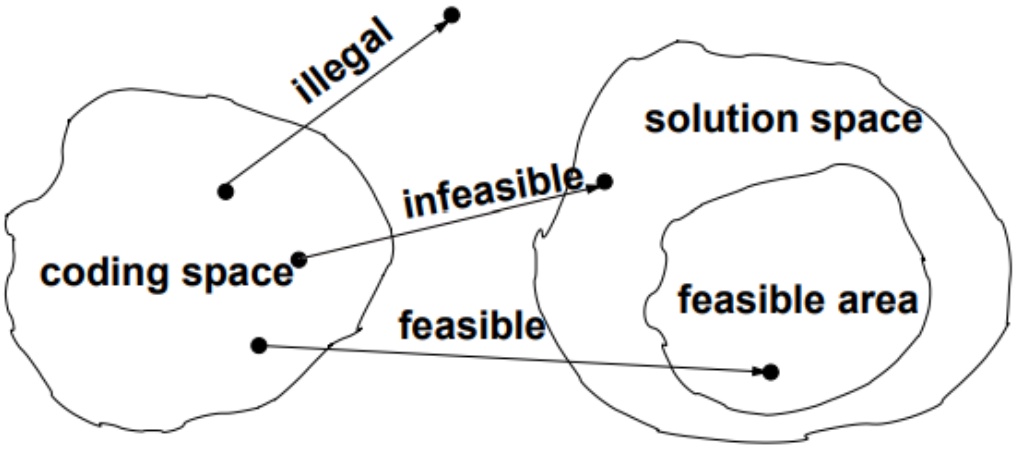

Figure 3.4. Feasibility and legality. (Cheng, Gen, & Tsujimura, 1996)

Operations perform in GA:

• Selection

• Crossover

• Mutation

Selection is the process by which individuals are chosen from the current population to serve as parents for the next generation. The selection process is based on the fitness values of the individuals, which are determined by the fitness function.

The goal of selection is to increase the frequency of individuals with high fitness values in the population, while decreasing the frequency of individuals with low fitness values. This is done through a process of probabilistic selection, in which individuals with higher fitness values are more likely to be selected as parents than those with lower fitness values.

Chromosome 1 1101100100110110

Chromosome 2 1101111000011110

There are several different selection strategies that can be used in a genetic algorithm, including:

Roulette wheel selection: This is a commonly used selection strategy in which individuals are selected with a probability proportional to their fitness values. The individuals are placed on a roulette wheel, with each individual occupying a slice of the wheel proportional to its fitness value. The wheel is spun, and the individual that the wheel lands on is selected as a parent.

Tournament selection: In this strategy, individuals are selected in pairs, and the individual with the higher fitness value is selected as a parent. The tournament size (i.e., the number of individuals in each tournament) is a parameter that can be adjusted to control the strength of selection.

Rank-based selection: This strategy assigns a rank to each individual in the population based on its fitness value, and the individuals are selected with a probability proportional to their rank. This can be useful when the fitness values are highly variable or noisy, as it places more emphasis on the relative ordering of the individuals than on their exact fitness values.

Once the parents have been selected, they undergo the crossover and mutation operations to produce offspring for the next generation. The selection process is then repeated for the next generation, and the genetic algorithm continues to iterate until a stopping criterion (e.g., a maximum number of generations) is reached.

Crossover is a genetic operator that is used to recombine the genetic material of two parent individuals to create offspring for the next generation. The crossover operation mimics the biological process of genetic recombination that occurs during sexual reproduction.

During crossover, two parent individuals are selected from the population based on their fitness values. The selected parents are then combined to create new offspring by exchanging portions of their genetic material (i.e., their chromosomes).

Chromosome 1 10111 | 10100110110

Chromosome 2 01011 | 11010011110

Offspring 1 10111 | 11010011110

Offspring 2 01011 | 10100110110

There are several different crossover strategies that can be used in a genetic algorithm, including:

Single-point crossover: A single point is chosen in the chromosomes of the parents, and the genetic material on either side of the point is exchanged to create the offspring.

Two-point crossover: This strategy is similar to single-point crossover, but two points are chosen in the chromosomes, and the genetic material between the two points is exchanged.

Uniform crossover: Each bit in the chromosomes of the parents is independently chosen to be included in the offspring with a probability of 0.5. This can result in more diverse offspring than the other crossover strategies.

The choice of the crossover process can have a significant impact on the performance of the genetic algorithm. Some strategies may be better suited to certain types of problems or certain characteristics of the population.

The crossover operation is typically followed by the mutation operation, which introduces random changes to the genetic material of the offspring. The mutation operation helps to introduce additional genetic diversity into the population, which can help to avoid premature convergence to suboptimal solutions.

Mutation is a genetic operator that introduces random changes to the genetic material of an individual in the population. The mutation operator mimics the biological process of genetic mutation, which can occur spontaneously in the DNA of an organism.

The mutation operator is used to introduce new genetic diversity into the population and prevent the genetic algorithm from becoming stuck in local optima. Without mutation, the algorithm may converge too quickly to a suboptimal solution, as the genetic material of the population is not subject to random changes.

During mutation, a randomly selected subset of the genetic material (e.g., a single bit or a small segment of a chromosome) is modified according to some probabilistic rule. The specific mutation strategy used will depend on the problem being solved and the characteristics of the population.

Original offspring 1 10111 10100110110

Original offspring 2 01011 11010011110

Mutated offspring 1 10111 11010011110

Mutated offspring 2 01011 10100110110

There are several different mutation strategies that can be used in a genetic algorithm, including:

Bit-flip mutation: A random bit in the binary string of the individual is flipped (i.e., changed from 0 to 1 or from 1 to 0).

Gaussian mutation: A random value is added to the current value of a gene in the individual, with the new value being drawn from a Gaussian distribution with mean 0 and a small standard deviation.

Inversion mutation: A small segment of the chromosome is inverted (i.e., the order of the bits is reversed).

The choice of mutation strategy will depend on the problem being solved and the characteristics of the population. It is also important to balance the rate of mutation with the rate of crossover, as too much mutation can lead to a loss of beneficial genetic material, while too little mutation can lead to premature convergence to suboptimal solutions.

Once the problem has been encoded in a chromosomal manner, and a fitness measure for distinguishing favorable solutions from unfavorable ones has been selected, the evolutionary process of finding solutions to the search problem can commence. This process involves the following steps:

Step 1 Initialization: The initial population of potential solutions is typically created randomly across the search space. Nonetheless, domain-specific knowledge or other information may be integrated to generate the initial population.

Step 2 Evaluation: After the population is initialized or when an offspring population is formed, the fitness values of the potential solutions are evaluated.

Step 3 Selection: Selection allocates more copies to solutions with better fitness values, implementing the survival-of-the-fittest mechanism. The main objective of selection is to favor better solutions over worse ones.

Step 4 Recombination: Recombination involves combining segments of two or more parental solutions to create new, potentially improved solutions (offspring). There are various ways to accomplish this, and successful performance depends on designing the recombination mechanism effectively. The key idea is that the offspring produced through recombination will be distinct from any individual parent, combining parental traits in a novel manner.

Step 5 Mutation: While recombination involves two or more parental chromosomes, mutation, occurring locally but randomly, introduces modifications to a solution. Again, there are numerous variations of mutation, but it generally entails one or more changes to an individual's traits. In essence, mutation performs a random walk in the vicinity of a potential solution.

Step 6 Replacement: The offspring population generated through selection, recombination, and mutation replaces the original parental population. Different replacement techniques such as elitist replacement, generation-wise replacement, and steady-state replacement methods are utilized in GAs.

Step 7 Repeat steps 2–6 until one or more stopping criteria are met. The evolutionary process continues iteratively until specific conditions are satisfied, indicating convergence or the achievement of desired solution quality.

The flow of these steps is illustrated in Figure 3.5. and the pseudo codes are given in Table 3.2.

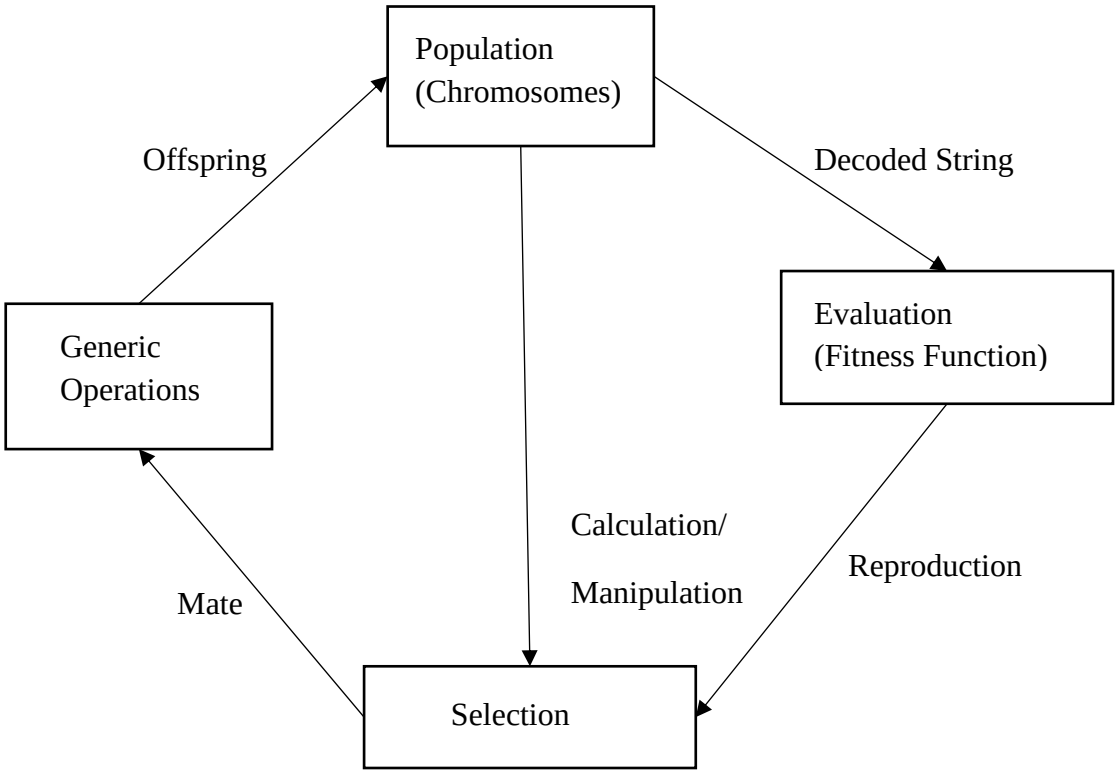

Figure 3.5. Genetic algorithm flow

Algorithm 1: Pseudo-code of the standard genetic algorithm

1: **Input:** N: Population size; Pc : Crossover rate; Pm : Mutation rate.

2: **Output:** Best Chromosome.

3: $t \leftarrow 0$

4: Initialize arbitrarily the initial population P(t)

5: **while** (not termination condition) **do**

6: Evaluate P(t) using a fitness function

7: Select P(t) from P ($t-1$)

8: Recombine P(t)

9: Mutate P(t)

10: Evaluate P(t)

11: Replace P ($t - 1$) by P(t)

12: $t \leftarrow t + 1$

13: **end**

Table 3.2. The steps of the standard GA

Holland (1975) presented a broad framework for Genetic Algorithms (GAs), which allows for various elaborations and variations, including different genetic operators and variable-sized populations. Grefenstette (1992), restricted the GA to a specific subclass defined by six parameters.

1) Population Size (N): The size of the population significantly impacts the performance and efficiency of GA's. When using very small populations, GAs tend to perform poorly due to the limited sample size for most hyperplanes. On the contrary, a larger population is more likely to include representatives from a greater number of hyperplanes, enabling a more informed search.

Consequently, employing a larger population size helps in avoiding premature convergence to suboptimal solutions. However, a larger population also entails more evaluations per generation, which can lead to a slower rate of convergence. In our current experiments, we considered population sizes ranging from 10 to 160 in increments of 10.

2) Crossover Rate (C): The crossover rate is the frequency at which the crossover operator is utilized. In each new population, $C * N$ population members undergo crossover.

A higher crossover rate leads to a faster introduction of new members into the population. However, if the rate is too high, high-performance members may be discarded at a pace that exceeds the improvements produced by selection. Conversely, if the crossover rate is too low, the search process might stagnate due to a lower exploration rate.

In this study, we examined 16 different crossover rates, ranging from 0.25 to 1.00 in increments of 0.05, to explore the effects on the GA's performance and convergence behavior.

3) Mutation Rate (M): Mutation is another operator that introduces variability into the population. After the selection process, each bit position of every member in the new population undergoes a random change with a probability determined by the mutation rate *M*. As a result

of this operator, almost *M * N * L* mutations occur per generation, where *L* is the length of each member.

A low mutation rate prevents any particular bit position from being permanently converged to a single value across the entire population. This helps maintain diversity and avoid premature convergence. On the other hand, a high mutation rate results in a more random search, which can be beneficial for escaping local optima.

In the current experiments, we explored eight values for the mutation rate, increasing exponentially from 0.0 to 1.0. This allowed us to study the impact of different mutation rates on the GA's performance and behavior.

4) Generation Gap (G): The generation gap determines the proportion of the population to be replaced during each generation. Specifically, *N * (1 - G)* structures from the current population P(t) are randomly chosen to survive unchanged in the next population *P (t + 1).*

A value of *G* = 1.0 indicates that the entire population is replaced in each generation. On the other hand, a value of *G* = 0.5 means that only half of the structures in the population continue to the next generation.

In the current experiments, we explored various values of *G* between 0.30 and 1.00 in increments of 0.10. By studying different generation gap settings, we aimed to analyze how the proportion of the population retained in each generation affects the GA's behavior and convergence characteristics.

5) Scaling Window (W): In the context of maximizing a numerical function *f(x)* using a GA, it is common to observe the performance value *u(x)* of a member x as $u(x) = f(x) - f_{min}$, where f_{min} is the minimum value that *f(x)* can attain within the existing search space. This operation ensures that the performance *u(x)* is positive, irrespective of the characteristics of *f(x).* However, in practice, f_min is often not known in advance, leading to an alternative definition u(x) = f(x) – f_{min} where $f(x_{min})$ represents the minimum value obtained from any structure evaluated so far.

Both definitions of *u(x)* can present a challenge in distinguishing good solutions, as it reduces the selection pressure towards better-performing structures. For example, let's assume $f(x_{min})$, = 0. After several generations, the current population may consist of structures x for which 105

< *f(x)* < 110. At this stage, no structure in the population exhibits a significantly deviating performance from the average. Consequently, the selection process stagnates, affecting the search efficiency.

To address this issue, we introduced a new parameter called f'_{min} with a predetermined value, for instance, 100, and rated each member against this standard. For example, if f(x_i) = 110 and f(x_j) = 105, then *u(x)* = f(x_i) – f'_{min} = 10, and *u*(x_j) = f(x_j)- f'_{min} = 5. This adjustment makes the performance of x_i appear twice as good as that of x_i.

During our experiments, we explored three scaling modes, based on the scaling window parameter *W*. If *W* = 0, we implemented scaling as follows: f'_{min} was set to the minimum *f(x)* observed in the first generation. For each subsequent generation, members with evaluations less than f_{min} were excluded from the selection procedure. We updated f'_{min} whenever all members in a given population had evaluations greater than f'_{min}. If 0 < *W* < 7, we set f_min to the least value of *f(x)* observed in the last *W* generations. A value of *W* = 7 indicated an infinite window, implying that no scaling was applied.

Through these experiments, we aimed to assess the impact of the scaling window on the performance of the GA and understand its influence on the selection process and search efficiency.

6) Selection Strategy (S): The experiments involved a comparison of two selection strategies. If *S = P, a pure selection procedure* was employed, where each structure in the current population is replicated proportional to its performance. On the other hand, if *S = E, an elitist strategy* was utilized. First, pure selection was carried out, and additionally, the elitist strategy ensured that the structure with the best performance always survives unchanged into the next generation.

The implementation of an elitist strategy helps preserve the best-performing structure throughout the evolution process, preventing it from being lost due to sampling error, crossover, or mutation. Without this strategy, there is a possibility that the best structure may disappear, leading to suboptimal results.

In our experiments, we labeled each particular GA by specifying its corresponding values for the parameters *N, C, M, G, W,* and *S*. By comparing the performance of GAs with different

selection strategies, we aimed to understand the influence of these strategies on the overall GA performance and the preservation of the best solutions.

Early research by De Jong (1993) provided valuable insights into parameter settings that have been widely utilized in various genetic algorithm implementations. Building upon De Jong's work, Grefenstette (1992) established the standard *GA* denoted as GA_s = GA (50, 0.6, 0.001, 1.0, 7, E). This standard GA configuration represents a specific combination of the six parameters *(N, C, M, G, W, S*).

The Cartesian product of the specified parameter ranges for *N*, *C*, *M*, *G*, *W*, and *S* results in a vast search space containing 2^{18} possible GA configurations. In certain cases, it becomes feasible to predict how changes in individual parameters can influence the GA's performance, under the condition that all other parameters remain fixed. (Grefenstette,1986).

4. COMPUTATIONAL STUDY

In this section, mathematical model for multiple facilities location problem is given and a genetic algorithm to solve this problem is explained.

4.1. MaxPSumPSumC Mathematical Formulation

The unweighted partial-sum dispersion problem which presented by Lei & Church (2015) 's was used. The model considers the system's objective function by selecting the smallest *K* partial sums (p-sums) that encompass the nearest *L* possibilities.

MaxPSumPSumC:

$$MAX\ z = K.t - \sum_{i=1}^{n} u_i \quad (1)$$

Subject to:

$t\text{-}u_i \le q_i \quad \forall\ i \in I \quad (2)$

$q_i = (L+1)s_i - \sum_{j \in I} z_{ij} \quad \forall i \in I \quad (3)$

$s_i\text{-}z_i \le y_i d_{ij} + M_i(2\text{-}y_i - y_i) \quad \forall i \in I, j \in I \quad (4)$

$$\sum_i y_i = p \quad (5)$$

$u_i \ge 0,\ z_{ij} \ge 0, \quad \forall i \in I, j \in I \quad (6)$

$y_i \in \{0,1\} \quad (7)$

Assume a finite, discrete set I of points representing potential facility sites. Consider the following notation:

I: is a finite set of points representing potential facility sites.

i, j: indices denote facility sites, where $i \in I, j \in I, I = \{1, 2, \ldots, n\}$.

d_{ij} : the distance that separates sites i and j. (the sum of the L + 1 shortest distances assigned to site i)

M_i: is a suitably large number and equal to L. $max_{j \in I} d_{ij}$ selected to be greater than any potential partial sum value for each i.

y_i : 1, if a facility is located at site i, and 0 otherwise.

u_i and z_{ij} are similar auxiliary variables.

z_{ij} : is non-zero value when j represents one of the $L + 1$ closest assignments to site i, and it takes zero otherwise. Similar to variable u_i, z_{ij} is also associated with the sum of the $L + 1$ closest assignments to site i.

It is important to consider the sum of the $L + 1$ closest facilities to i since it includes the zero self-distance from i to itself (i.e., $d_{ii} = 0$). In other words, the sum of the $L + 1$ smallest distances will be equivalent to the sum of distances to the L closest neighboring facilities.

The objective function, as expressed in Equation 1, aims to maximize the total sum of the $L + 1$ closest distance assignments to each facility site i, which mostly ensures that facilities are located in proximity to the nearest possibilities.

Equation 2 ensures that for each facility site i, the difference between the variable t and the auxiliary variable u_i is less than or equal to the value q_i. This constraint mostly ensures that the selected distance assignments are within the specified limits.

Equation 3 defines the relationship between the variables q_i, s_i, and z_{ij}. It mostly ensures that q_i, representing the sum of the smallest L + 1 distance assignments for site i, is influenced by

the values of s_i and z_{ij}, the latter being an auxiliary variable that indicates if a site *j* is among the L + 1 closest assignments to site *i*.

Equation 4 relates the variables s_i, z_i, y_i, d_{ij}, and M_i. It mostly ensures that the difference between s_i, and z_i, is constrained based on the binary variable y_i, which indicates the presence or absence of a facility at site *i*, and the separation distance d_{ij} between sites *i* and *j*. The parameter M_i, being a sufficiently large constant, helps enforce the constraint.

Equation 5 imposes a constraint on the binary variables y_i. It mostly ensures that the sum of all y_i values (indicating the presence of a facility at site *i*) is equal to a constant *p*.

Equations 6 specify non-negativity constraints for the auxiliary variables u_i, and z_{ij}, ensuring that they remain greater than or equal to zero.

Finally, Equation 7 denotes that the binary variables y_i can only take on values 0 or 1, ensuring that facilities are either present or absent at site *i*.

In this study, the unweighted partial-sum dispersion problem, as introduced by Lei & Church (2015), was employed. The objective of this problem is to optimize facility location selection by considering the system's objective function based on the smallest K partial sums (psums) that encompass the nearest *L* possibilities. The model utilized a set of finite, discrete points denoted by *I*, representing potential facility sites.

Node	**X**	**Y**	**Node**	**X**	**Y**
1	32	31	29	19	38
2	29	32	30	27	41
3	27	36	31	21	35
4	29	29	32	32	45
5	32	29	33	27	45
6	26	25	34	32	38
7	24	33	35	8	22
8	30	35	36	15	25
9	29	37	37	35	16
10	29	21	38	35	47
11	33	28	39	46	51
12	17	53	40	50	40
13	34	30	41	23	22
14	25	60	42	27	30
15	21	28	43	38	39
16	30	51	44	36	32
17	19	47	45	32	41

18	17	33	46	42	36
19	22	40	47	36	26
20	25	14	48	15	19
21	29	12	49	19	14
22	24	48	50	45	19
23	17	42	51	27	5
24	6	26	52	52	24
25	19	21	53	40	22
26	10	32	54	40	52
27	34	56	55	42	42
28	12	47			

Table 4.1. Swain dataset 55 nodes

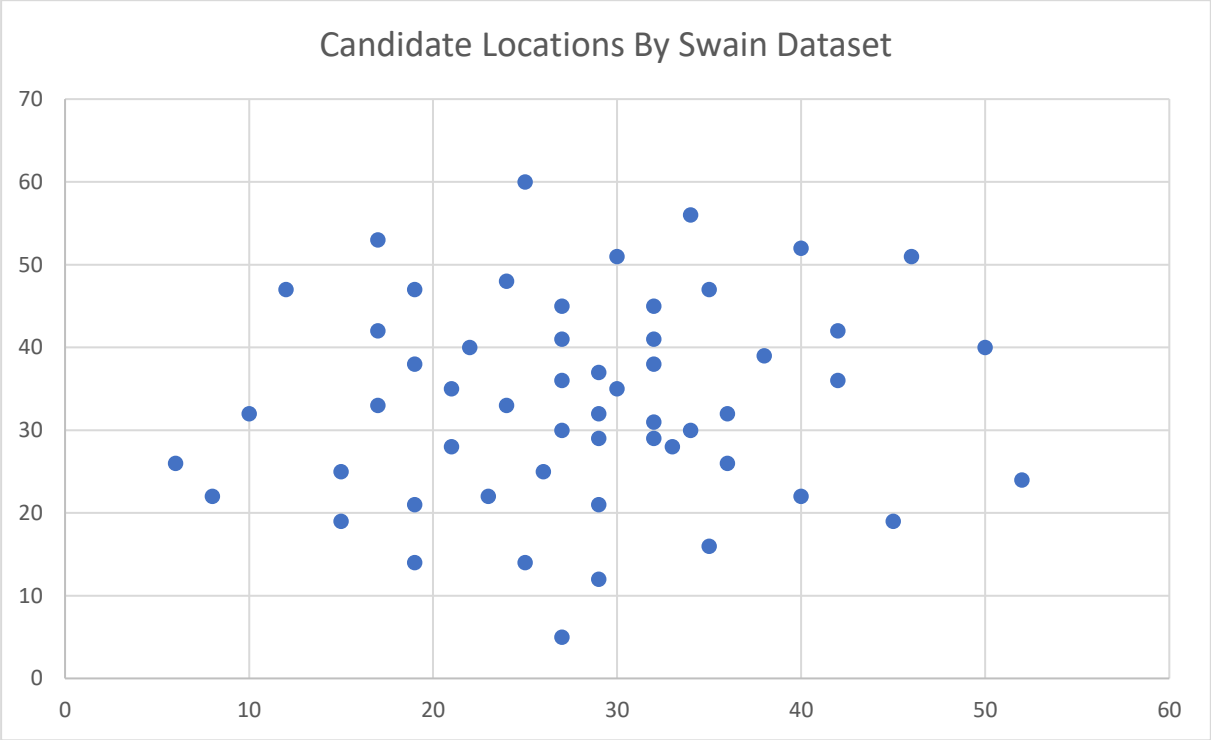

Figure 4.1. Site locations by Swain

4.2. Solution Algorithms

The 55-node test network by Swain as shown in Figure 4.1. and coordinates are listed in Table 4.1. is a widely used benchmark network in the field of facility location algorithms. It provides a standardized and representative set of input data to evaluate the performance and effectiveness of different algorithms in solving facility location problems. The network consists of 55 nodes representing demand points and candidate facility locations, along with the associated input parameters. Let's go through the step-by-step explanation of the inputs in this test network:

Node Coordinates:

Each node in the network is defined by its geographical coordinates, typically latitude and longitude. These coordinates determine the spatial placement of the nodes on a map.

Demand Points:

The test network includes a set of demand points, which represent locations where services or products are required. These could be customer locations or areas with specific demands.

Each demand point is associated with a specific demand quantity, indicating the level of service required.

Candidate Facility Locations:

The network also includes a set of candidate facility locations, which represent potential sites for establishing facilities.

Each candidate location is associated with a certain cost, representing the investment required to establish a facility at that location.

Distance Matrix:

To evaluate the distances between nodes, a distance matrix is provided. It contains the pairwise distances between each demand point and each candidate facility location.

The distances can be measured in terms of physical distance (e.g., Euclidean distance) or travel time between locations, depending on the specific problem context.

Capacity Constraint:

In many facility location problems, there is a capacity constraint associated with each candidate facility location. This constraint specifies the maximum amount of demand that a facility can serve.

The capacity constraint helps ensure that facilities are not overwhelmed with demand and can operate within their operational limits.

Objective Function:

The objective function defines the optimization goal of the facility location problem. It typically aims to minimize the total cost, which includes the facility establishment costs, transportation costs, and any other relevant costs.

The objective function can be customized based on the specific requirements of the problem, such as minimizing distance, maximizing coverage, or balancing facility loads.

By providing these inputs, the 55-node test network allows researchers and practitioners to compare the performance of different facility location algorithms and assess their effectiveness in solving real-world problems. The network captures the complexities and challenges of facility location decisions, including spatial considerations, demand quantities, cost factors, and capacity constraints. The step-by-step explanation of each input helps establish a common ground for evaluating algorithmic solutions and fostering advancements in the field of facility location optimization.

Let's assume the following inputs for the genetic algorithm code:

POPULATION_SIZE = 80

NUM_FACILITIES = 3

K = 1

L = 1

CANDIDATE_SITES = [(1, 1), (2, 3), (5, 4), (8, 1), (4, 2)]

Generating the initial population: The generate_random_solution function is called POPULATION_SIZE times to generate the initial population. The initial population will contain 80 random solutions, where each solution is a list of 3 candidate sites chosen randomly. For example: Initial population: [[(2, 3), (8, 1), (5, 4)], [(1, 1), (5, 4), (2, 3)], ...]

Run the genetic algorithm loop: The loop runs for a single generation (range 1). This means the following steps are performed only once.

Evaluate the fitness of each solution: For each solution in the population, the evaluate_fitness function is called. It calculates the fitness of a solution as the sum of the K worst-case facility-based sum of distances. The distances between facilities are computed using the distance function. The fitness values for the population are calculated and printed. Example output: Facility 0 partial sum of 1 smallest distance: 2.23606797749979 Facility 1 partial sum of 1

smallest distance: 3.1622776601683795 Facility 2 partial sum of 1 smallest distance: 2.23606797749979 Facility distances in descending order: [3.1622776601683795, 2.23606797749979, 2.23606797749979] Fitness values: [3.1622776601683795, 2.23606797749979, 2.23606797749979, ...]

Select the fittest solutions for reproduction: The fittest solutions are selected by finding the maximum fitness value and collecting all the solutions with that fitness value into the fittest_solutions list. In this case, there is only one fittest solution because K is set to 1 and there is only one facility-based sum of distances. Example output: Fittest solutions: [[(2, 3), (8, 1), (5, 4)], ...]

Generate the next generation of solutions through crossover and mutation: The next generation is generated by performing crossover and mutation operations. The loop runs for POPULATION_SIZE - num_fittest_solutions iterations, which in this case is 80 - 1 = 79 iterations.

Select two parent solutions randomly from the current population.

Perform crossover by randomly selecting a pivot point and combining the two parent solutions at that point.

Mutate the child solution with a low probability (10% chance) by randomly selecting one of the facilities and replacing it with a different candidate site. • Add the generated child solution to the next_generation list. Example output: Mutated Solution: [(1, 1), (8, 1), (5, 4)]

Add the fittest solutions to the next generation: The fittest solutions are added to the next_generation list obtained from step 5.

Replace the current population with the next generation.

The current population is replaced with the next_generation. Get the fittest solution in the final population: The fittest solution is obtained by finding the maximum fitness value in the final population and selecting the corresponding solution. Example output: Fittest solution: [(5, 4), (8, 1), (1, 1)]. Therefore, the final solution obtained for the given input is [(5, 4), (8, 1), (1, 1)].

5. RESULTS AND DISCUSSIONS

Running time comparison of Genetic Algorithm, Greedy Algorithm and Exact Solution in milliseconds as in Table 5.1.:

p	***K***	***L***	**Genetic times**	**Greedy times**	**Exact times**
5	1	1	15,8193	0,0937	374
5	1	2	8,9355	0,0951	546
5	1	4	9,0767	0,5095	109
5	2	1	8,6139	0,1144	390
5	2	2	122,2246	8,6854	827
5	2	4	8,6426	0,0937	218
5	5	1	9,0350	0,1121	280
5	5	2	9,3668	0,1197	171,6
5	5	4	13,0440	0,1350	200
7	1	1	11,7503	0,1653	1310
7	1	2	8,5733	0,7110	2200
7	1	3	8,5687	0,1511	4740
7	1	6	9,1335	0,1893	561
7	2	1	8,5406	0,1987	1170
7	2	2	8,9979	0,1499	3320
7	2	3	11,5348	0,1389	7630
7	2	6	11,4465	0,1506	686
7	3	1	11,7293	0,1493	1326
7	3	2	11,8771	0,1490	3510
7	3	3	11,2756	0,1467	7780
7	3	6	12,1608	0,1447	610
7	7	1	11,6110	0,1494	1280
7	7	2	11,7844	0,8957	6536
7	7	3	11,5277	0,2132	18210
7	7	6	11,4684	0,1511	4945

Table 5.1. Running times

The table presents the execution times (in milliseconds) for the Genetic algorithm, Greedy algorithm, and the Exact method in obtaining their respective solutions. The "Genetic times" column shows the time taken by the Genetic algorithm to find its approximate solutions, the "Greedy times" column displays the time required by the Greedy algorithm to generate its approximate solutions, and the "Exact times" column indicates the time needed by the Exact method to compute the optimal solutions.

lt is evident that both the Genetic and Greedy algorithms offer significantly lower execution times compared to the Exact method. This illustrates the computational efficiency of both heuristic approaches in quickly finding approximate solutions. However, it is worth noting that

the Greedy algorithm generally has slightly shorter execution times compared to the Genetic algorithm.

The heuristic algorithms (Genetic and Greedy) offer fast and efficient solutions, making them suitable for large-scale problems or situations where quick results are essential. On the other hand, the Exact method guarantees optimal solutions but requires more computational resources and time.

Comparation of objective functions as in Figure 5.1.:

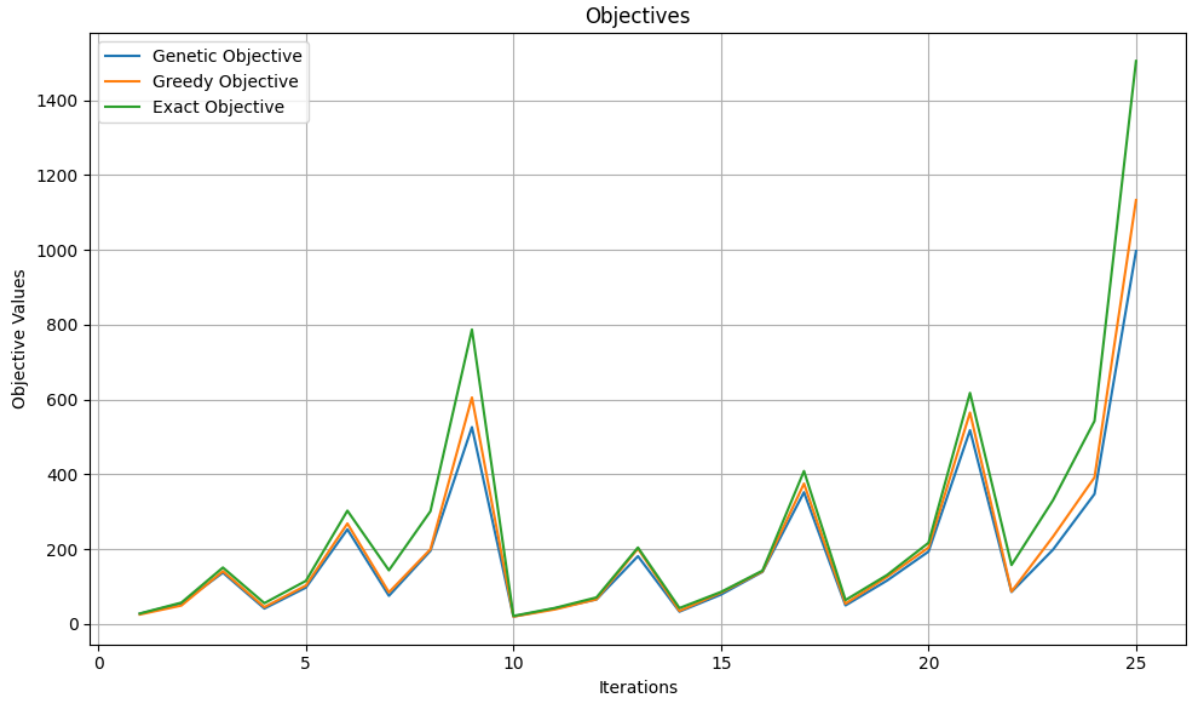

Figure 5. 1. Comparation of GA, Greedy and Exact algorithm

p	***K***	***L***	**Genetic objective**	**Greedy objective**	**Exact Objective**	**Relative Error (Genetic)**	**Relative Error (Greedy)**
5	1	1	26,9961	25,1137	27,6586	2,40	7,86
5	1	2	54,5098	48,9655	56,7275	3,91	34,37
5	1	4	136,8973	140,8319	150,556	9,07	19,18
5	2	1	41,0658	44,7939	55,3172	25,76	1,79
5	2	2	96,9901	102,5099	114,951	15,62	10,85
5	2	4	252,6433	268,31539	302,623	16,52	4,64
5	5	1	75,0028	84,247	143,239	47,64	12,62
5	5	2	195,2381	199,3626	301,107	35,16	6,55

5	5	4	525,9906	605,3467	787,146	33,18	11,04
7	1	1	19,6397	19,3618	21,095	6,90	41,42
7	1	2	41,2024	38,6273	42,4726	2,99	77,65
7	1	3	65,2728	65,5681	70,6179	7,57	71,98
7	1	6	180,9237	201,17086	204,138	11,37	32,77
7	2	1	32,6402	34,5656	42,19	22,64	33,47
7	2	2	78,1791	83,2766	85,2902	8,34	43,06
7	2	3	139,0700	140,2151	142,292	2,26	40,79
7	2	6	351,8906	375,0239	408,647	13,89	14,71
7	3	1	49,3265	53,9791	63,5676	22,40	30,23
7	3	2	115,6626	125,4256	130,538	11,40	29,59
7	3	3	193,2494	203,9843	216,674	10,81	28,10
7	3	6	517,7671	565,0346	617,577	16,16	12,12
7	7	1	85,2728	86,3679	157,415	45,83	4,51
7	7	2	198,4505	233,1866	330,634	39,98	0,90
7	7	3	347,3468	391,7073	542,635	35,99	2,37
7	7	6	997,0099	1133,1531	1505,27	33,77	5,30

Table 5.2. Relative error

Figure 5.2. Relative error

Objective functions were taken from the average of ten times run. Relative error indicates that the Genetic or Greedy Algorithm solution is how far from the optimal solution. The comparation of relative error values are shown in Figure 5.2. and the relative error for each iteration was given in Table 5.2. For example, in the first scenario, the Genetic Algorithm

solution is approximately 28.12% higher, and the Greedy Algorithm solution is approximately 6.13% far from the optimal solution.

Magnitude of percentage value indicates how close or far the heuristic solutions (Genetic and Greedy Algorithms) are from the optimal solution. Larger magnitude percentages represent solutions that are farther from the optimal, while smaller magnitude percentages indicate solutions that are closer to the optimal.

Overall, both algorithms show varying performance across different scenarios. In some cases, they may close to the optimal solution, while in others, they fall short. Heuristic algorithms like Genetic and Greedy are designed for faster, approximate solutions and may not always guarantee optimal results. However, these algorithms provide a balance between solution quality and computational efficiency, making them valuable tools in tackling complex optimization problems. Decision-makers can choose the appropriate algorithm based on their priorities, either seeking a relatively better solution quickly (heuristic algorithms) or ensuring an exact optimal solution at the cost of increased computational effort (exact methods). Comparison of selected sites as shown in Table 5.3.:

p	***K***	***L***	**Genetic Sites**	**Greedy Sites**	**Branch and bound Sites**
5	1	1	[5 17 22 35 50]	[4 17 20 38 51]	[12 24 39 51 52]
5	1	2	[14 27 29 35 40]	[3 16 21 35 40]	[12 35 39 51 52]
5	1	4	[12 28 32 43 51]	[6 23 27 39 47]	[12 24 39 51 52]
5	2	1	[14 21 52 42 50]	[17 21 24 53 54]	[12 24 39 51 52]
5	2	2	[20 22 27 38 51]	[24 27 35 37 39]	[12 35 39 51 52]
5	2	4	[3 16 28 44 49]	[10 18 27 35 55]	[12 35 39 51 52]
5	5	1	[7 13 14 22 48]	[28 35 40 50 51]	[12 35 39 51 52]
5	5	2	[4 20 28 48 55]	[21 55 31 40 49]	[12 35 39 51 52]
5	5	4	[27 36 37 38 39]	[27 28 42 49 52]	[12 35 39 51 52]
7	1	1	[2 6 7 8 14 19 40]	[12 14 28 40 48 49 53]	[24 27 28 40 42 50 51]
7	1	2	[10 15 31 36 38 50 55]	[12 16 21 27 35 49 52]	[24 27 28 40 42 50 51]
7	1	3	[2 4 7 10 30 40 49]	[5 10 23 27 49 53 54]	[12 14 35 39 48 50 52]
7	1	6	[4 13 20 22 46 51 52]	[12 16 20 26 46 48 51]	[14 24 27 35 40 51 52]
7	2	1	[6 9 12 23 28 44 53]	[10 12 25 28 43 50 52]	[24 27 28 40 42 50 51]
7	2	2	[14 16 24 25 41 43 53]	[6 24 27 28 37 47 54]	[2 27 28 35 40 50 51]]
7	2	3	[2 24 32 34 46 45 51]	[21 26 33 35 39 47 54]	[12 14 35 39 48 50 52]
7	2	6	[17 28 34 37 45 48 52]	[16 21 24 35 39 50 52]	[14 24 27 35 40 51 52]
7	3	1	[15 17 23 27 29 51 53]	[14 28 32 37 49 51 55]	[24 27 28 40 42 50 51]
7	3	2	[11 16 22 36 39 52 51]	[21 26 28 38 42 52 55]	[1 14 28 35 39 51 52]
7	3	3	[14 15 17 27 36 40 51]	[5 21 29 32 35 53 51]	[14 28 35 39 47 51 52]
7	3	6	[4 11 26 34 37 50 52]	[4 10 14 24 32 33 52]	[12 14 21 24 39 51 52]
7	7	1	[1 18 21 41 42 43 52]	[14 22 39 40 48 51 53]	[3 14 24 28 39 51 52]

7	7	2	[11 14 15 30 33 38 49]	[16 26 39 48 49 50 54]	[2 14 24 28 39 51 52]
7	7	3	[4 9 23 25 24 49 50]	[20 23 26 46 49 52 54]	[14 24 28 39 50 51 52]
7	7	6	[14 17 24 25 36 43 55]	[17 20 23 27 41 52 54]	[12 14 24 35 39 51 52]

Table 5.3. Selected sites

The success of the Genetic and Greedy Algorithms in a significant portion of the scenarios highlights their effectiveness as approximate optimization techniques for solving the given problem. Although they may not always guarantee an exact optimal solution, their ability to come close to the optimal distances in most cases makes them valuable tools for tackling complex optimization problems efficiently.

6. CONCLUSIONS

After conducting the test scenarios, we can confidently state that both the genetic algorithm and the greedy algorithm have proven to be successful in finding solutions that are close to the exact solutions for the facility location problem involving obnoxious facilities. These heuristic algorithms showcased their effectiveness in tackling the complexity of this NP-hard problem.

The genetic algorithm demonstrated its strength in exploring a wide range of potential facility sites while considering multiple criteria and constraints. By iteratively refining the solutions and maintaining diversity in the population, it effectively achieved well-balanced solutions that prioritize harm reduction during the optimization process.

On the other hand, the greedy algorithm, with its simple and efficient approach of making locally optimal choices, provided competitive solutions for the facility location problem. It was particularly effective in situations where immediate gains were crucial.

As part of future work, a valuable enhancement could be to add a radius constraint to the model. This constraint would limit the distance between obnoxious facilities and sensitive areas, ensuring that potential facility sites maintain a safe distance from residential neighborhoods or environmental conservation zones. Integrating this constraint into both algorithms would further improve solution quality and minimize potential harm to communities and the environment.

In conclusion, the genetic algorithm and the greedy algorithm have demonstrated their capabilities in providing effective approximate solutions to the challenging facility location problem for obnoxious facilities. These heuristic approaches offer a balance between solution quality and computational efficiency, making them valuable tools for addressing complex optimization problems in real-world scenarios.

REFERENCES

Abdel-Basset M., Abdel-Fatah L., & Sangaiah, A.R. (2018). Metaheuristic algorithms: A comprehensive review. In Computational intelligence for multimedia big data on the cloud with engineering applications. Elsevier, 185–231.

Akgün V., Erkut E., & Batta R. (2000). On finding dissimilar paths. Eur J Oper Res 121:232–246.

Akgün V., Parekh A., Batta R., & Rump C. (2007). Routing of a hazmat tuck in the presence of weather systems. Comput Oper Res 34:1351–1373.

Alumur S., & Kara B. (2007). A new model for the hazardous waste location-routing problem. Comput Oper Res 34:1406–1423.

Amirgaliyeva, Z., Mladenovîc, N., Todosijeviîc, R., & Uroševiîc, D. (2017). Solving the maximum min-sum dispersion by alternating formulations of two different problems. European Journal of Operational Research, 260(2), 444–459.

Batta, R., & Chiu, S.S. (1988). Optimal Obnoxious Paths on a Network: Transportation of Hazardous Materials. Oper Res 36(1):84–92.

Bell, M. (2007). Mixed Routing Strategies for Hazardous Materials: Decision-Making Under Complete Uncertainty. Spat Econ 6:253–265.

Berman, O., Drezner, Z., & Wesolowsky, G. (2007). The transfer point location problem. Comput Oper Res 34:1374–1388.

C. Toregas and C. ReVelle, "Optimal Location under Time or Distance Constraints," Papers of the Regional Science Association, XXVIII, 1972.

Cappanera, P., Gallo, G., & Maffioli, F. (2004). Discrete facility location and routing of obnoxious activities. Discrete Appl Math 133:3–28.

Carotenuto, P., Giordani, S., Ricciardelli, S., & Rismondo, S. (2007) A tabu search approach for scheduling hazmat shipments. Comput Oper Res 34:1328–1350.

Castillo, J.E. (2004). Route optimization for hazardous materials transport. Enschede, The Netherlands.

Church, R. L., & Drezner, Z. (2022). Review of obnoxious facilities location problems Computers & Operations Research, 138, 105468.

Church, R. L., & Revelle, C. (1974). The maximal covering location problem. Papers of the Regional Science Association, 32:101–118.

Church, R.L., & Garfinkel, R.S. (1978.) Locating an obnoxious facility on a network. Transportation Science 12, 107–118.

Church, R.L., & Cohon, J.L. (1976). Multiobjective location analysis of regional energy facility sitting problems. Report prepared for the US energy research and development administration (BNL 50567).

Colebrook M., Guti. J, & Sicilia J. (2005). A new bound and an O (mn) algorithm for the undesirable 1-median problem (Maxian) on networks. Comput Oper Res 32:309–325.

Curtin, K.M., & R. L. Church. (2006). A family of location models for multiple type discrete dispersion. Geographical Analysis 38: 248–270.

Curtis, S.A. (2003). The classification of greedy algorithms. Science of Computer Programming, 49:125– 157.

De Jong K. (1993). Genetic algorithms are not function optimizers, in: L.D. Whitley (Ed.), Foundations of Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, pp. 5–17.

Dîaz-Bânez J.M., Gomez F., & Toussaint G.T. (2005). Computing shortest paths for transportation of hazardous materials in continuous spaces. J Food Eng 70:293–298.

Drezner, Z., & Wesolowsky, G.O. (1980). A maximin location problem with maximum distance constraints. AIIE Transactions 12 (3), 249–252.

Erkut E. & Neuman S. (1991). Comparison of four models for dispersing facilities. INFOR Canadian Journal of Operational Research and Information Processing 29: 68–86.

Erkut E., & Ingolfsson A. (2005). Transport risk models for hazardous materials: revisited. Oper Res Lett 33:81–89.

Erkut E., & Neuman S. (1989). Analytical models for locating undesirable facilities, European Journal of Operational Research 40 275-291.

Erkut E., & Verter V. (1998). Modeling of transport risk for hazardous material. Oper Res 46(5): 625–642.

Erkut E., &Alp O. (2007). Designing a network for hazardous materials shipments. Comput Oper Res 34(5):1389–1405.

Farahani R.Z., & Hekmatfar M. (2009). Facility location: concepts, models, algorithms and case studies. Springer.

Fernandez J., Fernandez P., & Pelegrin B. (2000). A continuous location model for siting a non-noxious undesirable facility within a geographical region. Eur J Oper Res 121:259–274.

Giannikos I. (1998). A multi-objective programming model for locating treatment sites and routing hazardous wastes. Eur J Oper Res 104:333–342.

Goldman, A.J., & Dearing, P.M. (1975.) Concepts of optimal location for partially noxious facilities. ORSA/TIMS National Meeting, Chicago.

Gopalan R., Kolluri K.S., Batta R., Karwan M.H. (1990). Modeling equity of risk in the transportation of hazardous materials. Oper Res 38(6):961–973.

Grefenstette J. J., (1986). Optimization of Control Parameters for Genetic Algorithms. IEEE Transaction on Systems, Man and Cybernetics, vol. 16, no. 1, pp. 122–128.

Grefenstette J.J., (1992). Genetic algorithms for changing environments, in: Parallel Problem Solving from Nature 2, PPSN-II, Brussels, Belgium, September 28–30, pp. 139–146.

Holland J. H., (1992). Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press.

Katz M.J., Kedem K., Segal M. (2002). Improved algorithms for placing undesirable facilities. Comput Oper Res 29:1859–1872.

Kuby M.J. (1987). Programming models for facility dispersion: the p-dispersion and maxisum dispersion problems. Geographical Analysis 19:315–32.

Kumar M., Husian M., Upreti N., & Gupta D. (2010). Genetic algorithm: Review and application. International Journal of Information Technology and Knowledge Management, 2(2):451–454.

Lambora A., Gupta K. & Chopra K. (2019). Genetic Algorithm- A Literature Review, International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 2019, 380- 384.

Lamini, C., Benhlima, S., & Elbekri, A., (2018). Genetic algorithm based approach for autonomous mobile robot path planning, in: Procedia Computer Science. doi:10.

Lei, T.L., & Church, R.L. (2015). On the unified dispersion problem: efficient formulations and exact algorithms. Eur. J. Oper. Res. 241(3), 622–630.

Marianov V., & ReVelle C. (1998). Linear, no-approximated models for optimal routing in hazardous environments. J Operl Res Society 48:157–164.

McGinnis L.F., & White J.A. (1978). A single facility rectilinear location problem with multiple criteria. Transport Sci 12:217–231.

Mehrez A., Sinuany-Stern Z., & Stulman A. (1986). An enhancement of the Drezner–Wesolowsky algorithm for single facility location with maximin of rectilinear distance. J Oper Res Soc 37:971–977.

Melachrinoudi E. (1999). Bicriteria location of a semi-obnoxious facility. Comput Oper Res 37: 581–593.

Melachrinoudis, E. & MacGregor Smith, J. (1995). An o (mn2) algorithm for the maximin problem in e 2. Operations Research Letters, 18, 25–30.

Moon I.D., & Chaudhry S.S. (1984). An analysis of network location problems with distance constraints. Management Science 30, 290-307.

P.L. van den Berg, J.T. van Essen, and E.J. Harderwijk. Comparison of static ambulance location models. Under review, 2014.

Pisinger D. (2006). Upper bounds and exact algorithms for p-dispersion problems. Comput Oper Res 33:1380–1398.

Pisinger D., (2006). Upper bounds and exact algorithms for p-dispersion problems," Comput. Oper. Res., vol. 33, no. 5, pp. 1380–1398.

Rakas J., Teodorovic D., & Kim T. (2004). Multi-objective modeling for determining location of undesirable facilities. Transport Re D9:125–138.

Rana, R. & Garg, D. (2014). Algorithm for Obnoxious Facility Location Problem. International Journal of Advancements in Technology, Vol.5, No. 4, Pp. 96-106.

Ratick S., & White A. (1988). A risk sharing model for locating noxious facilities. Environment Plan 165–179.

Rodriguez J.J., Garcia G.C., Mûnoz Pêrez J Mêrida Casermeiroa E. (2006). A general model for the undesirable single facility location problem. Oper Res Lett 34(4):427–436.

Rogers G.O. (1998). Sitting potentially hazardous facilities: What factors impact perceived and acceptable risk? Landsc Urban Plan 39:265–2.

Shier, D.R. (1977). A minmax theorem for p-center problems on a tree. Transportation Science 1l, 243-252.

Ting, S.S. (1984). A linear-time algorithm for maxisum facility location on tree networks, Transp. Sci. 18, 76–84.

Van den Berg P.L., Van Essen J.T., Van Harderwijk E.J. (2016). Comparison of static ambulance location models. Comparison of static ambulance location models. In 2016 3rd International Conference on Logistics Operations Management (GOL), pages 1–10, May 2016.

Wayman M.M., & Kuby M. (1994). Proactive optimization: General framework and a case study using a toxic waste location model with technology choice. International symposium on locational decisions, ISOLDE VI, Lesvos and Chios, Greece.

Wayman M.M., Kuby M. (1994). Proactive optimization: General framework and a case study using a toxic waste location model with technology choice. International symposium on locational decisions, ISOLDE VI, Lesvos and Chios, Greece.

White, J., & Case, K. (1974). On covering problems and the central facility location problem. Geographical Analysis, 281, 1974.

Zhang J., Hodgson J. & Erkut E. (2000). Using GIS to assess the risk of hazardous materials transport in networks. Eur J Oper Res 121:316–329.

Zografos K.-G., & Samara S.S.A. (2007). Combined location-routing model for hazardous waste transportation and disposal. Transportation Research Record 1990.

APPENDIX

Appendix A

```
import random
import time
# Constants
POPULATION_SIZE = 80
NUM_FACILITIES = 5
K = 1
L = 1
CANDIDATE_SITES = [(32,31),(29,32),(27,36),(29,29),(32,29),
(26,25),(24,33),(30,35),(29,37),(29,21),(33,28),(17,53),(34,30),
(25,60),(21,28),(30,51),(19,47),(17,33),(22,40),(25,14),(29,12),
(24,48),(17,42),(6,26),(19,21),(10,32),(34,56),(12,47),(19,38),
(27,41),(21,35),(32,45),(27,45),(32,38),(8,22),(15,25),(35,16),(35,47),(46,51),
(50,40),(23,22),(27,30),(38,39),(36,32),(32,41),(42,36),(36,26),(15,19),
(19,14),(45,19),(27,5),(52,24),(40,22),(40,52),(42,42)]
```

```
def distance(point1, point2):
    return ((point1[0] - point2[0]) ** 2 + (point1[1] - point2[1]) ** 2) ** 0.5
def evaluate_fitness(solution):
        facility_distances = []
    for i in range(NUM_FACILITIES):
        distances = []
        for j in range(NUM_FACILITIES):
            if i != j:
                dist = distance(solution[i], solution[j])
                distances.append(dist)
        distances.sort()
        facility_distances.append(sum(distances[:L]))
    facility_distances.sort(reverse=True)
    return sum(facility_distances[:K])

def generate_random_solution():
    solution = random.sample(CANDIDATE_SITES, NUM_FACILITIES)
    return solution

def mutate_solution(solution):
        mutated_solution = solution.copy()
    facility_index = random.randint(0, NUM_FACILITIES - 1)
    mutated_solution[facility_index] = random.choice(CANDIDATE_SITES)
    return mutated_solution

def crossover_solutions(solution1, solution2):
```

```
    pivot = random.randint(1, NUM_FACILITIES - 1)
    return solution1[:pivot] + solution2[pivot:]

def main():
    population = [generate_random_solution() for _ in range(POPULATION_SIZE)]

    start_time = time.perf_counter()

    for generation in range(100):
        fitness_values = [evaluate_fitness(solution) for solution in population]

        fittest_solutions = [population[i] for i in range(POPULATION_SIZE) if fitness_values[i]
== max(fitness_values)]
        num_fittest_solutions = len(fittest_solutions)
            next_generation = []
        for _ in range(POPULATION_SIZE - num_fittest_solutions):
            parent1, parent2 = random.sample(population, 2)

            child = crossover_solutions(parent1, parent2)

            if random.random() < 0.1:
                child = mutate_solution(child)

            next_generation.append(child)

                next_generation += fittest_solutions
```

```
        population = next_generation

    end_time = time.perf_counter()  # Stop tracking the solution time
    solution_time = (end_time - start_time) * 1000  # milliseconds

    fittest_solution = max(population, key=lambda solution: evaluate_fitness(solution))
    objective_value = evaluate_fitness(fittest_solution)

    fittest_solution_with_numbers = [(i+1, CANDIDATE_SITES.index(point)+1, point) for i, point in enumerate(fittest_solution)]
    print("Fittest solution:")
    for num, index, point in fittest_solution_with_numbers:
        print(f"{index}: {point}")

    print(f"Objective value: {objective_value}")
    print(f"Solution time: {solution_time} milliseconds")

if __name__ == "__main__":
    main()
```

Appendix B

```
import random
import time
NUM_FACILITIES = 5
K = 1
L = 1
```

```
CANDIDATE_SITES =
[(32,31),(29,32),(27,36),(29,29),(32,29),(26,25),(24,33),(30,35),(29,37),(29,21),(33,28),(17,53
),(34,30),(25,60),(21,28),(30,51),(19,47),(17,33),(22,40),(25,14),(29,12),(24,48),(17,42),(6,26
),(19,21),

(10,32),(34,56),(12,47),(19,38),(27,41),(21,35),(32,45),(27,45),(32,38),(8,22),(15,25),(35,16),
(35,47),(46,51),(50,40),(23,22),(27,30),(38,39),(36,32),(32,41),(42,36),(36,26),(15,19),(19,14)
,(45,19),(27,5),(52,24),(40,22),(40,52),(42,42)]


def distance(point1, point2):
    return ((point1[0] - point2[0]) ** 2 + (point1[1] - point2[1]) ** 2) ** 0.5


def evaluate_fitness(solution):
    facility_distances = []
    for i in range(NUM_FACILITIES):
        distances = []
        for j in range(NUM_FACILITIES):
            if i != j:
                dist = distance(solution[i], solution[j])
                distances.append(dist)
        distances.sort()
        facility_distances.append(sum(distances[:L]))
    facility_distances.sort(reverse=True)
    return sum(facility_distances[:K])


def generate_greedy_solution():
    chosen = []
    remaining = CANDIDATE_SITES.copy()
    while len(chosen) < NUM_FACILITIES:
```

```
        max_total_dist = -1
        best_candidate = None
        for candidate in remaining:
            total_dist = sum(distance(candidate, chosen_facility) for chosen_facility in chosen)
            if total_dist > max_total_dist:
                max_total_dist = total_dist
                best_candidate = candidate
        chosen.append(best_candidate)
        remaining.remove(best_candidate)
    return chosen


def main():
    start_time = time.perf_counter()


     solution = generate_greedy_solution()
    end_time = time.perf_counter()
    solution_time = (end_time - start_time) * 1000
        objective_value = evaluate_fitness(solution)
    print(f"Fittest solution: {solution}")
    print(f"Objective value: {objective_value}")
    print(f"Solution time: {solution_time} milliseconds")
if __name__ == "__main__":
    main()
```