Heuristics for Facility Location Problems

A thesis submitted during 2022 to the University of Hyderabad in partial fulfillment of the award of a **Ph.D. degree** in School of Computer and Information Sciences

by

Edukondalu Chappidi

School of Computer and Information Sciences
University of Hyderabad
P.O. Central University, Gachibowli
Hyderabad – 500 046
Telangana, India

December 2022

CERTIFICATE

This is to certify that the thesis entitled **"Heuristics for Facility Location Problems"** submitted by **Edukondalu Chappidi** bearing **Reg. No. 16MCPC05** in partial fulfillment of the requirements for the award of **Doctor of Philosophy** in **Computer Science** is a bonafide work carried out by him under my supervision and guidance.

This thesis is free from plagiarism and has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

The student has the following publications before submission of the thesis for adjudication and has produced evidence for the same in the form of acceptance letter or the reprint in the relevant area of his research:

1. **Edukondalu Chappidi**, Alok Singh. "Discrete differential evolution-based solution for anti-covering location problem". *Proceedings of the 10th International Conference on Soft Computing for Problem Solving (SocProS 2020), Advances in Intelligent Systems and Computing,* ***1392****: 607-620, 2021, Springer*. Work reported in this paper appears in **Chapter 2**.
2. **Edukondalu Chappidi**, Alok Singh and Rammohan Mallipeddi. "Intelligent optimization algorithms for disruptive anti-covering location problem". To appear in *Proceedings of the 19th International Conference on Distributed Computing and Intelligent Technology (ICDCIT 2023), Lecture Notes in Computer Science, 2023, Springer*. Work reported in this paper appears in **Chapter 3**.
3. **Edukondalu Chappidi**, Alok Singh. "Evolutionary approaches for the weighted anti-covering location problem". To appear in *Evolutionary Intelligence, Springer*. Work reported in this paper appears in **Chapter 3**.

4. **Edukondalu Chappidi**, Alok Singh. "An evolutionary approach for obnoxious cooperative maximum covering location problem". *Applied Intelligence,* ***52****: 16651–16666, 2022, Springer*. Work reported in this paper appears in **Chapter 4**.

and has made the presentation in the following conference:

1. 10^{th} International Conference on Soft Computing for Problem Solving (SocProS 2020), December 18-20, 2020, Indore, India.

Further, the student has passed the following courses towards fulfillment of coursework requirement for Ph.D.:

Course Code	**Name**	**Credits**	**Pass/Fail**
CS 801	Data Structures and Algorithms	4	Pass
CS 802	Operating Systems and Programming	4	Pass
AI 810	Metaheuristic Techniques	4	Pass
AI 852	Learning & Reasoning	4	Pass

(Prof. Alok Singh)
Supervisor
School of Computer and Information Sciences
University of Hyderabad
Hyderabad – 500 046, India

(Prof. Chakravarthy Bhagavathi)
Dean
School of Computer and Information Sciences
University of Hyderabad
Hyderabad – 500 046, India

DECLARATION

I, **Edukondalu Chappidi**, hereby declare that this thesis entitled **"Heuristics for Facility Location Problems"** submitted by me under the guidance and supervision of **Prof. Alok Singh** is a bonafide research work which is also free from plagiarism. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma. I hereby agree that my thesis can be deposited in Shodganga/INFLIBNET.

A report on plagiarism statistics from the University Library is enclosed.

Date :

Name: Edukondalu Chappidi

Signature of the Student:

Reg. No.: 16MCPC05

Signature of the Supervisor:

Abstract

Facility location problems are concerned with locating facilities over a set of potential locations subject to some constraints so that a given objective function is optimized (minimized or maximized). The objective function can be based on factors such as cost of locating facilities, distance between facilities and customers, number of facilities that needs to be opened, service time, waiting time, coverage or a combination of these factors. There are many application domains such as locating public facilities, commercial facilities, factories, power plants, ware-houses. To tackle these real-world problems, various facility location models have been developed as a result of abstraction.

In this thesis, we have worked on three facility location models and variants thereof, viz. anti-covering location problem (ACLP), obnoxious cooperative maximum coverage location problem (OCMCLP), reliability p-median problem (RpMP). We choose these models because these models are under-studied despite several real-world applications. In addition, choice of latter two models is also governed in part by their highly complex nature which pose a challenge to anyone trying to solve them. We have considered six $\mathcal{NP}$-hard facility location problems in this thesis. Out of these, first three are based on ACLP and its variants. The fourth problem deals with location of obnoxious facilities under cooperative coverage. The last two problems are the p-median facility location problems which consider the reliability and fault tolerance issues. These problems have many practical applications in diverse fields such as locating garbage dump yards, nuclear power plants, chemical plants, telecommunication equipments, franchise outlets, liquor stores, ATMs, military defense units, DNA sequence matching, forest management, supply chain design, disaster management. As these problems are $\mathcal{NP}$-hard, applicability of exact methods is limited to small size instances only, and one has to resort to heuristic approaches to tackle instances beyond a certain maximum size.

We have devised heuristic approaches based on genetic algorithm (GA), discrete differential evolution (DDE), and hyper-heuristics to address the considered facility location problems. In addition, we have also developed some problem-specific heuristics for use within these approaches. We have compared the performance of our proposed approaches with the state-of-the-art approaches available in the literature on the standard benchmark instances of the respective problems. Computational results show the efficacy of our proposed approaches. The proposed approaches can be easily extended to solve other related facility location problems. The ideas presented in the thesis can be used to develop heuristic approaches for other combinatorial optimization problems also.

To my dear parents,

Mr. Chappidi Kasulu and Mrs. Venkata Ramana

my dear wife and lovely sons,

Rachel, Paul and Ben

without their endless love, support and encouragement, this would not have been possible.

Acknowledgements

It has been my passion to pursue Ph.D and I am deeply thankful to many people who are part of this journey and helped me realize this dream. I would like to express my deep appreciation to all of them.

Above all I am grateful to God Almighty for His blessings and grace on my life without which nothing would have been possible.

Next, I express my sincere gratitude towards my supervisor **Prof. Alok Singh** for his constant support and guidance throughout this Ph.D. I believe it was my destiny to work under his esteemed guidance to undertake this course while learning the proper research in the truest sense. His impeccable knowledge and astute feedback helped me face some of the toughest research problems in the field of combinatorial optimization and never give up in the process in spite of innumerable technical difficulties. I am greatly indebted to him for his valuable time, patience, and insightful guidance offered to me. His attention to detail and striving towards perfection in each considered task are something I want to imbibe in my journey ahead.

Next, I would like to thank my doctoral review committee (DRC) members, **Dr. Sobha Rani T.** and **Dr. Rajendra Prasad Lal** for their probing queries, feedback, and suggestions which helped me to enhance the quality of my research from various perspectives.

I take this opportunity to thank the Dean of the School **Prof. Chakravarthy Bhagvati** for providing all the necessary facilities to pursue my research work. I would also like to thank other faculty members and staff of the school for their support. I am thankful for the unstinting support that I received from the research infrastructure and the effervescent ambiance of the University. Due credit to the University for building a research oriented School of Computer & Information

Sciences (SCIS), a library rich in a wide range of research books & articles, and most importantly a healthy campus atmosphere.

I am thankful to my senior lab mate **Dr. Venkatesh Pandiri** who had helped me in the initial stages of my Ph.D right from making me comfortable in the lab to introducing me to the scenic and pleasant locations in the University campus. I also want to make a special mention about my other senior lab mate **Dr. Gaurav Srivastava** for his friendship and engaging technical discussions in the lab. I am thankful to my fellow lab mates (**Kasi Vishwanath, Danish, Sebanti and Preeti**) for stimulating discussions, providing me company for tea/snacks and for all the fun we had together. I am thankful to all my Ph.D colleagues in SCIS for their support and encouragement.

I would like to make a special mention about my dear friend and labmate **Mallikarjun** whom we lost due to COVID-19. I shared a close, brotherly bond with him and miss his pleasant smile and presence in the lab.

I am grateful to a number of researchers in my field who shared their research data and answered my queries regarding their problem formulations and proposed approaches, especially to **Ms. Preeti Ravindranath, Prof. Sohail Chaudhry, Prof. Averbakh Igor, Dr. Alcaraz Javier, Dr. Maria Albareda-Sambola**.

I want to thank my friends and extended family who played an important role in my life and helped me in my journey so far. I fondly remember my first guru and our primay school teacher **Late. Mr. G Murali Krishna** sir and my inspiration in secondary school **Mr. K Edwin** sir. I want to make a mention of my uncle **Mr. Cheeti Nagendra Kumar**, who first planted the thought of doing Ph.D in my mind many years ago. I express my sincere gratitude towards **Mr. Selvam MPT** for being my mentor and my support system ever since we first met. I want to thank my all-weather friend **Mr. Sai Kumar** for the encouragement and being there for me.

Finally, I could not have achieved anything in life without the unconditional and constant support of my family. I would like to dedicate this thesis to my parents **Mr. Chappidi Kasulu** and **Mrs. Venkata Ramana** who have been my inspiration and instilled right values and taught me to work hard from a young age. I would like to thank a very special person, my wife, **Rachel** for her continued and unfailing love,

support and understanding during my pursuit of Ph.D. I am thankful to my precious kids who are the stress busters of my life, **Paul and Ben**. I want to thank my elder brothers **Venkat and Srinivas** and their families for their affection, patience and encouragement. I want to make a special mention of my in-laws **Mr. Vijaya R and Mrs. Rathnamma** for their love and support.

Edukondalu Chappidi

Contents

List of Figures

List of Tables

LIST OF TABLES

Chapter 1

Introduction

In the field of Operations Research (OR), facility location is an important branch that continues to attract researchers over the past many decades. Identifying the best locations for facilities is one of the trickiest and most important decisions to be made in a project due to the high competition among the companies. These choices are especially crucial for companies because of the substantial expenditures associated with locating and relocating the facilities. Additionally, the placements of a company's facilities with respect to other facilities and its clients play a role in its capacity to effectively produce and sell its goods or provide high-quality services. Facility location models find their applications in both public and private sectors for locating a wide variety of facilities such as schools, fire stations, police stations, banking facilities, restaurants, medical services, swimming pools, vehicle service centers, shopping malls, hazardous and waste material disposal.

It is observed that in the facility location problems related to the application fields such as food service, retail trade and refuelling, the number of facilities that needs to be located is often fixed. On the other hand, in the facility location problems arising out of other application fields, the aim is to find the number of facilities to be located and also to decide the locations of the facilities. It is crucial to determine the precise number of facilities that must be located and the best location for each one since locating too many or too few facilities or wrongly locating the facilities will lead to decreased performance and increased costs. In industrial applications, deploying too many facilities will result in costs that are higher than desired. If fewer facilities are established, the level of customer service may deteriorate, which may decrease consumer loyalty. Despite using the appropriate number of facilities, improper facility location will lead to sub-par performance while increasing the overall cost. In the application fields such as health

care and social assistance where emergency service facilities like ambulances, fire stations and police stations are to be located facility location is very crucial. Locating too few of such emergency facilities or mislocating them can result in adverse consequences in the emergency scenarios [3].

A facility location problem's characteristics are mainly dependent on the type of objectives considered while satisfying the given constraints on the facilities being located. The application where the facility location problem is applied will determine the optimization criteria. For example, in the case of locating private facilities like an industrial plant or a retail store the objectives are to maximize the overall profit and to gain higher market share from the existing competitors while minimizing the capital expenditure. On the other hand, in the applications that involve locating public facilities such as schools and hospitals the primary goals are to establish effective systems with probity of service at minimum cost possible. But for emergency facilities such as ambulances and fire stations, the objectives are defined based on criteria such as minimizing the total weighted distance from all the customers to their nearest facilities or minimizing the distance or travel time of the farthest user from the facility to receive the service or a combination of both the just mentioned criteria [4]. Usually, facility location problems are concerned with either minimizing the cost of locating and operating the facilities considering the facilities provide complete coverage to all the customers or maximizing the customer coverage that can be achieved given a fixed number of facilities [5].

Facility location problems are concerned with locating facilities over a set of potential locations subject to some constraints so that a given objective function is optimized (minimized or maximized). The objective function can be based on factors such as cost of locating facilities, distance between facilities and customers, number of facilities that needs to be opened, service time, waiting time, coverage or a combination of these factors. As mentioned earlier, there are many application domains such as locating public facilities, commercial facilities, factories, power plants, ware-houses. To tackle these real-world problems, various facility location models have been developed as a result of abstraction. Any facility location problem typically involves two decisions where the first decision establishes the location of facilities, and the second one governs the allocation of customers to facilities in order to meet their respective service demands. Hence, the facility location problems are also called as location-allocation problems [6].

Different facility location models have been formulated based on criteria such as the number of candidate facilities, objective function, the problem's solution space. Considering the

problem's solution space, there are three facility location models, viz. continuous facility location models, discrete facility location models, and network facility location models.

Continuous facility location models have a continuous space generally determined by a plane's coordinates. In these models, we can locate facilities anywhere on the given plane or in the specified region having infinite number of possible locations. New facilities are located in the continuous space taking into consideration the already located facilities. Summary of location theory in continuous space is given in [7], whereas the latest works on continuous facility location problems are provided in [8, 9]. Survey of various approaches for solving the continuous facility location problems is given in [10].

In a discrete facility location problem, the solution space has a finite number of locations for locating facilities. Given a set of possible locations for facilities and a set of constraints, these problems are concerned with selecting a subset of locations where facilities can be located while satisfying the given constraints so that a given cost function can be optimized. A detailed survey of discrete facility location problems is provided in [11]. In [12], Drezner mentioned various models of discrete facility location problems while discussing their applications. In [13], Basu et al. provide a survey on applying metaheuristic methods for handling discrete facility location problems.

Continuous facility location models differ from discrete facility location models in that they need to select a distance or cost function that determines the distance or cost of a point from other points in the space. Example distance functions are Manhattan distance function, Euclidean distance function etc. On the other hand, in discrete facility location models, given any pair of points the actual travel distance or cost between them may be used. Discrete models are slower as compared to the continuous models due to the large amount of travel distance data they require as input and the pre-computations thereof to find the shortest distances. But they are more precise than continuous models as they use exact distance values as against the approximate distance values used in continuous models due to the distance function used [10].

Network facility location problems can be considered as a generalization of discrete facility location problems [14]. In these problems, given a set of nodes and edges connecting those nodes, facilities can be located either on the nodes or along the edges connecting the nodes. These problems are concerned with locating facilities anywhere on the network as just mentioned subject to some constraints while optimizing the given cost function. Based on the problem under consideration, the network can be either a network of air transport or a network of road transport or river transport, etc. [15].

In location problems, generally various types of objective functions are taken into consideration. Here are a few examples of objective functions that are frequently used: Minimizing the cost of overall set-up, minimizing the total number of facilities that are located, minimizing the maximum distance a customer has to travel to receive the service, minimizing the waiting time before a customer receives service after reaching the facility, minimizing the time taken to serve each customer etc. In the past two decades or so, the study of multi-objective facility location problems is also gaining popularity [16, 17].

In the field of facility location, covering problems are fascinating models as they are more suitable to solve several real-world problems particularly those which involve locating emergency facilities. Church and ReVelle first introduced the concept of coverage objective in [4]. In these problems, a customer is considered as covered if the distance between the customer and its closest facility is less than a pre-specified value called coverage distance or critical distance. This model of coverage where only a single facility decides whether a customer is covered or not is called individual coverage model. There are also other type of coverage models where a group of facilities together provide service to customers. The models where facilities cooperate and serve customers are called cooperative coverage models [18]. Covering problems find their applicability in solving several real-world problems including but not limited to locating parks, police stations, hospitals, post offices, radar installations, banks, shopping malls and dump-yards [19]. [20, 21, 22] give more information on the covering problems. One can refer to [2, 18, 23, 24] for more details about cooperative coverage models.

Generally covering problems involve locating facilities as near to the customers as possible. But, when the facilities to be located are undesirable like dump yards, prisons, nuclear power plants, sewage treatment plants etc., people want these facilities as far from their locations as possible. Models that deal with locating such undesirable facilities are called obnoxious facility location models. A review of facility location problems when the facilities are obnoxious is provided in [25]. The reviews of recent obnoxious models are presented in [26, 27].

There is a rich literature of facility location models and several variants of them that are considered and tackled using various techniques [16, 19, 28, 29, 30, 31, 32, 33]. In spite of location theory being regarded as an old field, several of these models are being used to solve many real-world problems making it a fascinating field of study. To understand more about facility location models, interested readers are requested to refer to [3, 12, 34, 35, 36, 37, 38, 39, 40]. Since nearly all variants of facility locating problems come under the category of $\mathcal{NP}$-hard problems, the applicability of exact techniques is limited to instances of a certain maximum

size only. Hence, several authors in the past have devised heuristic and metaheuristic based approaches to solve the facility location problems [10, 13, 41, 42, 43, 44]. Heuristics are intuitive methods that find feasible solutions in less time by making use of the structure of the problem at hand. No guarantee can be given about the solution quality when heuristics are applied. A heuristic generates better quality solutions when appropriate problem-specific knowledge is incorporated into the approach. Metaheuristic methods use problem-related information as components in their framework. The framework of a metaheuristic is independent of the considered problem. Many of the metaheuristic techniques are stochastic approaches and the resulting solution obtained after applying a metaheuristic depends on the generated values of several random variables. For the metaheuristics applied, they typically require the problem to be represented in a suitable form. Genetic algorithms [45, 46], differential evolution [47], tabu search [48, 49], ant colony optimization [50, 51], variable neighborhood search [52, 53], artificial bee colony algorithm [54], etc., are some of the most commonly used metaheuristic techniques. Metaheuristic-based approaches for solving facility location models have received a lot of attention of the researchers in the past [43, 44, 55, 56, 57]. It has been demonstrated in the literature that the approaches based on metaheuristics have outperformed problem-specific heuristics.

For any particular facility location problem, the solution methods often depend on the objective function of the problem and the constraints that need to be satisfied. For the facility location problems considered in this thesis, we have devised approaches based on genetic algorithm (GA), discrete differential evolution (DDE) and hyper-heuristics. In addition, we have developed some problem-specific heuristics for use with these approaches. In the next three sections, we provide overview of the approaches used in this thesis, viz. genetic algorithm (Section 1.1), differential evolution (Section 1.2) and hyper-heuristics (Section 1.3) respectively.

1.1 Overview of genetic algorithm

Genetic algorithm (GA) is a popular global optimization tool that models based on the principles of natural genetics and natural selection. It is one among the first few evolutionary algorithms that were proposed, and it is still widely used to solve optimization problems. John Holland introduced GA for the first time in 1960s with the intention of simulating evolutionary adaptation of the natural systems [45]. Later, GA was used to solve diverse problems including the ones related to optimization and search. Schema Theorem which was proved in [45] describes the

mechanism involved in the operation of a GA while giving a good theoretical explanation of it. According to the Schema Theorem, the number of schemata with fitness greater than average will rise with subsequent generations. The Schema Theorem functions as an analytical tool for GA and helps to determine which schema has a better probability of surviving the GA process. Holland also demonstrated GA's implicit parallelism using this theorem. Over the last many decades, there have been several variants of GA that were proposed to solve difficult optimization problems. GA has been a successful approach due to its characteristics such as simple structure, adaptability to a varied collection of problems and ability to arrive at good quality solutions because of its better search space exploration [58, 59, 60].

The terminology used in GA is extensively related to the biology, and hence, it helps to have a brief description of each of those terms in the backdrop of GA prior to diving into further details. The frequently used terms in GA and their descriptions are provided in Table 1.1.

Table 1.1: Frequently used terms in genetic algorithm parlance

Term	Explanation
Phenotype	A potential solution to the problem being considered
Chromosome or Genotype	The phenotype represented in a form on which GA can be applied
Gene	The smallest constituent in a chromosome and several genes together form a chromosome
Alleles	The set of values that a gene can be assigned with
Population	A collection of chromosomes participating in the evolution
Generation	A single pass from the current population to the next population
Fitness	A measure of chromosome's performance on the problem being considered
Evaluation	The process which gets a phenotype from the given genotype and determines its fitness
Phenotype Space	The space consisting of all potential solutions to the problem being considered
Genotype Space	The space consisting of all potential genotypes of the problem being considered

GA starts with a population of initial solutions or chromosomes for the problem being considered. These initial solutions can be generated either in a completely random manner or using some heuristics that make use of the problem-specific knowledge. By using the fitness function, each individual solution's fitness is evaluated which also gives a ranking of individual solutions to identify how one solution fares in comparison to other solutions in the population.

In most cases, the objective function of the problem is considered as the fitness function, but in some problems, the fitness function may be different from the objective function. After evaluating each solution in the initial population and assigning the corresponding fitness scores, GA works in an iterative manner. In every generation, some solutions are selected as parents according to a selection method. Generally, the selection method picks solutions with higher fitness to be part of the set of parent solutions due to the fact that better parents have a higher likelihood of having offspring who are even better. The genetic operators such as mutation and crossover are applied on the parent solutions to generate child solutions. Crossover is also referred to as recombination operator. It unites two or more parent solutions and creates one or more child solutions. As a result, the children produced by crossover have characteristics of each parent. Typically, just two solutions are recombined to create one or two child solutions during the crossover. Crossover is applied with a specific probability which is called crossover rate. Given a crossover rate of 0.7, about 70% of the child solutions will be produced using crossover. In the cases where crossover is not applied, a child solution is generated by creating an exact replica of the parent solution. Next, mutation is applied on the newly generated child solution whether crossover is applied or not. As part of the mutation, the newly generated child solution is made to go through some random changes. The probability with which mutation is applied is called mutation rate. Depending on the problem under consideration, mutation and crossover operators are applied either in a mutually exclusive manner or sequentially. Usually, both these operators are applied one after the other. In some cases, there may be a chance of crossover and mutation both being not applied on the parent solutions due to the crossover rate and mutation rate. In such scenarios, the newly generated child solution will be an exact replica of the parent solution. Once the required number of child solutions are generated, these newly generated child solutions compete with the solutions of the existing population to be part of the next generation. The population replacement policy determines the solutions which are to be included in the next generation, and then the next generation starts. This iterative process continues for as long as the termination criteria is not satisfied. There are different termination criteria that can be employed such as the number of generations or a predetermined amount of CPU time or the number of solutions produced or the number of iterations in a row without the best solution's quality improving.

Pseudo-code of basic GA is provided in Algorithm 1, where ps is the population size, q is the number of child solutions produced in each generation, and, p_c and p_m are crossover rate

Algorithm 1: Pseudo-code of basic GA

Input: Required parameters for GA and the considered problem instance
Output: Best solution returned by GA

```
Population ← φ;
for (i ← 1 to ps) do
    X_i ← Init_Solution();
    X_i.fitness ← Evaluate fitness of Solution(X_i);
    Population ← Population ∪ X_i;
while (the termination criteria is not satisfied) do
    Population' ← φ;
    for (i ← 1 to q) do
        parents ← Selection(Population);
        // Parents are selected based on a selection mechanism
        child_sol ← crossover(parents);
        // Crossover is applied as per p_c
        child_sol ← mutation(child_sol);
        // Mutation is applied as per p_m
        child_sol.fitness ← Evaluate fitness of child solutions;
        Population' ← Population' ∪ child_sol;
    Population ← evolution_policy(Population, Population');
return best_sol;
```

and mutation rate respectively. The function *Init_Solution()* produces an initial solution. We have explained various components of GA in subsequent subsections.

1.1.1 Representation of solutions

In GA, each solution in population of potential solutions is represented as a chromosome. The way a solution is represented is extremely important because it affects how chromosomes are manipulated to produce new chromosomes. As given in Table 1.1, actual solution to the problem is represented in phenotype and a genotype is an encoded form of the phenotype. The genetic operators of GA are applied on the genotype as part of the evolution. The solutions' genotypes should be represented in as natural way as possible such that the genotype space solution distribution is analogous to that of the phenotype space. The selected representation method should be able to represent all the solutions by completely avoiding the redundancy or by keeping it to minimum possible if it can not be avoided. If multiple genotypes correspond to the same phenotype or in other words if a single solution is represented by more than one chromosome, that representation method has redundancy. A representation method with redundancy leads to a larger genotype space for the associated phenotype space. This results in larger search space that

GA has to explore. Consequently, it could lead to poor GA performance [61]. So, the solution representation has a significant impact on how well GA performs on a given problem.

In the conventional GA, solutions are encoded using binary format often referred to as bit string representation. In the binary format, a chromosome consists of an array of binary digits 0s and 1s. The solution representation in binary format is most suitable for problems involving subset selection like the Knapsack problem where each index corresponds to a specific item. In the chromosome, if the value at i^{th} index is 1, it means that the corresponding item is present in the subset. On the other hand, if the value at i^{th} index is 0, it means that the corresponding item is not present in the subset. For example, consider a Knapsack problem having six objects and a solution represented in binary format as [0 1 0 0 1 1]. The given solution indicates that the second, fifth and sixth objects are in the subset. The given binary array [0 1 0 0 1 1] is the genotype for the phenotype [2 5 6] which is the original solution for the problem. For the permutation based optimization problems like TSP, the binary format of solution representation may not be suitable. Integer solution representation suits more for the permutation based problems to achieve better performance using GA. There are other solution representations like random-key encoding, real-valued representation etc., that can be applied depending on the problem under consideration.

1.1.2 Selection mechanisms

GA employs a selection mechanism for selecting the solutions that will participate in breeding and produce new higher quality solutions. A selection mechanism aims to increase the quality of solutions in the population by making sure the solutions with higher fitness have higher probability of reproducing. Different selection mechanisms differ from each other in terms of selection pressure and the degree of randomization employed in choosing the parent set. As a result, the selection mechanism plays a vital role in balancing exploration and exploitation. The success of GA depends on the selection mechanism employed. Various selection mechanisms are discussed in the literature [62]. Some of the most commonly used selection mechanisms are: fitness proportionate selection [45], ranking selection [63] and binary tournament selection [64].

1.1.2.1 Fitness proportionate selection

Holland [45] first introduced the fitness proportionate selection mechanism. As part of this mechanism, the probability of selecting an individual solution i is calculated as the ratio of i's

fitness value to the fitness sum of all the solutions in the population. Considering there are n solutions in the population and f_i gives the fitness of i^{th} solution, we can write the probability of selection of i^{th} solution as

$$prob_i = \frac{f_i}{\sum_{t=1}^{n} f_t}$$

According to this selection mechanism, a solution with higher fitness has a better probability of being chosen as a parent several times. From the population of solutions, a particular solution is selected using sampling methods such as stochastic universal sampling, roulette wheel approach. In both these sampling methods, each solution of the population is mapped to a distinct non-overlapping sub-interval of [0, 1] based on its selection probability. Consider an example with a population of three solutions having the following selection probabilities: 0.1, 0.4, 0.5. We can map these three solutions to following sub-intervals: [0, 0.1], (0.1, 0.5] and (0.5, 1.0] respectively. As part of the roulette wheel method, a random number in the range of [0, 1] is generated. In whichever of the three sub-intervals the generated random number falls, the corresponding solution is selected to be part of the set of parent solutions. This procedure is iteratively repeated till the required number of solutions are added to the set of parent solutions.

The fitness proportionate selection mechanism has the following two key limitations:

- In the starting phases of GA, there is high fitness gap among candidate solutions in the population. So, within few generations, the solutions with high fitness can take over the whole population resulting in the premature convergence of GA.
- After certain number of generations, all the solutions in the population may have similar fitness values which results in the roughly same selection probabilities for all solutions. Such a scenario makes the fitness proportionate selection mechanism ineffective.

1.1.2.2 Ranking selection

To overcome limitations of fitness proportionate selection mechanism, ranking selection was introduced [63]. In the ranking selection mechanism each solution in the population is assigned a rank based on its relative fitness as opposed to the absolute fitness. Individual solutions are sorted based on their fitness values and ranks are assigned. Considering $rank_i$ is the rank of i^{th} solution, its probability of selection $prob_i$ is calculated as

$$prob_i = \frac{rank_i}{\sum_{t=1}^{n} rank_t}$$

1.1.2.3 Tournament selection

In the tournament selection mechanism, only a subset of solutions of limited size are considered as the sample for selection, as opposed to the entire population of solutions. A subset of solutions of size *k* are randomly selected from the population. From among these *k* solutions, the solution having best fitness is chosen for reproduction, either based on a probabilistic method or in a deterministic manner. The tournament has to be conducted for several rounds to obtain the requisite number of parent solutions. This selection mechanism is known as binary tournament selection when only two solutions are selected to participate in the tournament, i.e., *k*=2. Binary tournament selection has a similar selection pressure as that of ranking selection though it is computationally more efficient than the ranking selection [64]. As part of the probabilistic binary tournament selection, two solutions are chosen uniformly at random from the population. Between the two selected solutions, the one with better fitness is selected as a parent with a given probability, p_{bt}. The inferior solution is selected as parent with the remaining probability, $1 - p_{bt}$.

1.1.3 Crossover

Crossover also known as the recombination operator, combines the genetic information of two or more parent solutions and generates child solutions. It is predicated on the notion that pairing together two good parents could result in a child of even higher quality. The offspring that arise from crossover may occasionally be worse than the parent solution, but repeated applications of crossover will result in solutions of better quality. While designing the crossover operator care should be taken so that it recombines the data pertinent to the considered problem. Additionally, if the two parents involved in the crossover are almost identical, then the child solution must likewise resemble the parents. This requirement is known as similarity requirement. The proper design of crossover operator as per the solution representation while making use of the problem specific knowledge is crucial to GA's success. We discuss some commonly used crossover operators in the subsequent subsections.

1.1.3.1 1-point crossover

Holland [45] introduced the 1-point crossover to be used in GA. Even though it can be applied to all forms of solution representations, it is most commonly applied in the case of binary and integer solution representations. Given a solution string of length n, the 1-point crossover

randomly selects a position in the range $[0, n-1]$. Then, the portions of the two parents are swapped from that point onwards thereby generating two child solutions. 1-point crossover is illustrated in Figure 1.1.

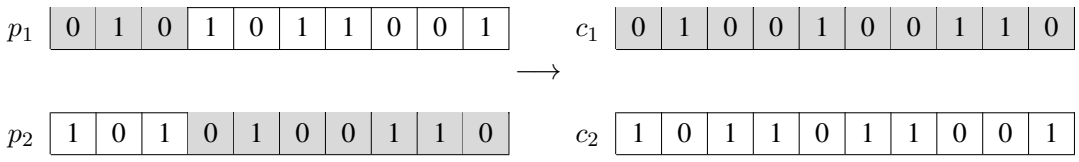


Figure 1.1: Illustration of 1-point crossover

By considering N crossover points instead of 1, we can implement an N-point crossover. Using the N crossover points, each parent is divided into $N+1$ different segments. By swapping every alternate segment between the two parents, we can generate two child solutions.

1.1.3.2 Uniform crossover

In the uniform crossover [65], each location in the child solution is separately considered. For each location, a uniform random number is generated within the range [0, 1] and is compared with the given probability value p. If the generated random number is less than or equal to p then the gene value from the same location in the first parent is copied to the child solution, otherwise the gene value from the second parent is copied. By switching the two parents' roles, the second child is produced. Uniform crossover is illustrated through an example in the Figure 1.2, where the probability value, p is taken as 0.6.

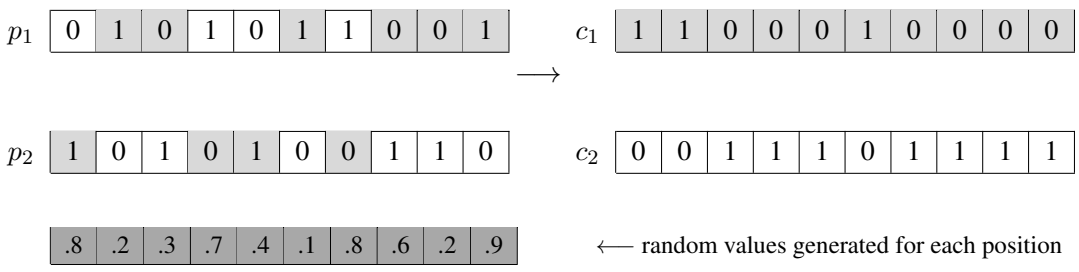


Figure 1.2: Illustration of uniform crossover

1.1.4 Mutation

Mutation operator helps in maintaining diversity in the population. It reduces the possibility of premature convergence, by avoiding a situation where the population's solutions become too similar. Additionally, it helps in exploring new areas of the search space. Just like in crossover, the solution representation mechanism employed by GA greatly influences the way mutation operator works. There are many mutation operators in the literature, which are designed as per the solution representation mechanism.

1.1.4.1 Bitwise mutation

Bitwise mutation is appropriate when solutions are represented as sequence of binary digits. After considering each bit in the solution separately, it is inverted with small uniform probability. Figure 1.3 provides an illustration of bitwise mutation with an example.

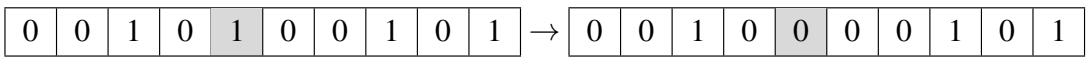


Figure 1.3: Illustration of bitwise mutation

1.1.4.2 Random reset mutation

Random reset mutation can be considered as an extension of bitwise mutation and it is suitable for integer solution representation. After considering each position in the solution, a different value from the list of possible values is assigned to it as per the given probability. Consider the illustration of random reset mutation given in Figure 1.4 with the possible values for a gene from the set {1, 2, 3, 4, 5, 6 }.

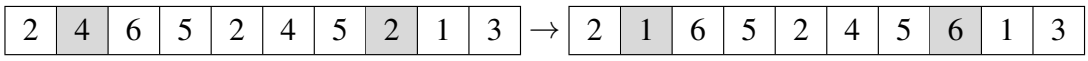


Figure 1.4: Illustration of random reset mutation

1.1.5 Population evolution models

The choice of solutions for the next generation is governed by the population evolution model. Generational model and steady-state model are the two major population models [66].

1.1.5.1 Generational model

In generational model, in each generation number of child solutions produced is equal to the population size. After each generation, the newly generated child solutions replace the whole population of solutions. Due to this kind of replacement policy, sometimes the newly generated population of solutions may be worse than the current generation solutions. Hence, some modifications have been introduced such as an elitist strategy which retains either the overall best solution or a number of good solutions from the current population and passes them to the next generation.

1.1.5.2 Steady-state model

In each generation, the steady-state model generates only a small fraction of solutions of the population size. The newly produced child solutions replace the same numbers of solutions in the current population as per a replacement policy. In this model, a newly generated child solution is considered for replacement into the population only if it is unique compared to the existing population members, otherwise it is discarded. This feature of the steady-state model makes sure that the population is free from duplicates thereby avoiding premature convergence. Typically in the steady-state model, only one child is produced per generation and if it is unique compared to the population members then it will take the place of worst solution in the population. Another replacement strategy is to select a solution that is most similar to the newly generated solution and replace it with the new solution.

1.2 Overview of differential evolution

Differential evolution (DE) is another population based metaheuristic technique inspired by natural evolution. It was proposed by Storn and Price in 1995 [47]. Originally, DE was proposed for solving continuous optimization problems in which the chromosomes are floating-point numbers [47, 67]. In the case of unknown problems, the initial population of solutions can be produced in a completely random manner. If any preliminary solutions are available for the problem, initial solutions can be generated by extending them by introducing random deviations which are normally distributed. Like several other evolutionary algorithms, DE also employs similar computational steps. However, to generate a new solution, it follows a completely different method. In this method, two distinct solutions from the population are randomly

selected and their weighted difference is added to a third distinct solution which is also selected randomly, giving a new child solution. Considering S_i, S_j, and S_k as three unique solutions which are randomly selected from the population and a user defined weighting factor w, we can mathematically present the step of generating a new child solution as:

$$S_P = S_i + w(S_j - S_k)$$

Owing to its simplicity, generic nature, and robustness, DE has become quite popular. As a result, there are numerous variations of basic DE in the literature that differ in the strategies used, such as the number of solutions included in perturbation, the type of mutation and crossover operators employed. A detailed survey of differential evolution is provided in [68].

As the traditional DE was developed for continuous optimization problems, it can not be used for solving discrete optimization problems. Tasgetiren et al. devised a novel discrete differential evolution (DDE) in [69, 70], which deals with solutions having discrete values. In the DDE approach, the solutions from the population are considered one after the other. The solution currently being considered is referred to as the target solution. Mutation can be applied either on the best solution or on a randomly selected candidate solution or on the current target solution [71] to produce a mutant solution. After the mutation, crossover is applied with a predefined probability considering the mutant and the target solution as parent solutions producing a trial solution. After the crossover, the fitness of the resulting trial solution is compared with that of the target solution. Following a selection mechanism, the trial solution may replace the target solution to be part of the population for next generation or is discarded.

Algorithm 2 provides the pseudo-code of discrete differential evolution. Here ps is the size of the population and the function *Init_Solution()* produces an initial solution.

DDE differs from other evolutionary techniques in selecting parent solutions for the crossover operator. In other techniques both parent solutions are selected from the population. On the other hand, in DDE, one parent solution is from the population, whereas the other parent is the resulting solution after mutating another solution in the population. Generally, either the best solution in the population or a randomly selected solution from the population is considered for perturbation and gives the second parent solution for crossover. Thus, one participating member of the crossover or recombination operator is typically a diversified solution. It has advantages, such as better exploration of search space and thereby mitigating premature

convergence, and better ability to produce diverse child solutions as one of the parents is always a perturbed solution.

Algorithm 2: Pseudo-code of DDE

Input: Required parameters for DDE and the considered problem instance
Output: Best solution returned by DDE

```
for (i=1 to ps) do
    X_i ← Init_Solution();
best_sol ← best solution among X_1, X_2, ... X_i, ..., X_ps;
while (the termination criteria is not satisfied) do
    for (i=1 to ps) do
        Mutant ← Mutate(X);
        // X is either randomly selected from the population or
           the best solution
        Trial_sol ← Crossover(Mutant, X_i);
        // X_i is the target solution & Trial_sol is the trial solution
        if (Trial_sol is better than X_i) then
            X_i ← Trial_sol;
            if (X_i is better than best_sol) then
                best_sol ← X_i;
return best_sol;
```

1.3 Overview of hyper-heuristics

In the field of discrete optimization, to solve any $\mathcal{NP}$-hard problem, usually researchers tend to develop heuristic or metaheuristic methods that make use of the problem specific knowledge. Even for problems under the same domain, the heuristic or metaheuristic methods require significant changes depending on the nature of the problem under consideration to be able to generate solutions of good quality in viable computational times. It is also demonstrated that the quality of solutions created by methods that properly combine different low-level heuristics outperforms those generated by each individual low-level heuristic [72, 73]. Hence, there is a need to develop approaches that can be used for solving problems across domains without incorporating deep problem specific knowledge to generate solutions of better quality by properly combining the low-level heuristics [74]. Hyper-heuristics are a suitable alternative solution methods as compared to the heuristics or metaheuristics due to their ability to adapt to the specifics of the problem instance under consideration while combining several low-level heuristics. A hyper-heuristic fundamentally differs from a metaheuristic in that the search space

for a hyper-heuristic is the set of knowledge-poor, easily implementable low-level heuristics, whereas for a metaheuristic the search space is the set of feasible solutions to the considered problem [75].

Given their generality in addressing problems, hyper-heuristics have received increased interest from the research community in the last 10 years or so. For the first time, Denzinger et al. [76] introduced the term hyper-heuristic to describe a method that combines some artificial intelligence based approaches to prove theorems in an automated manner. Later in [77], hyper-heuristics were defined as heuristics that can select the most appropriate heuristics from a set of low-level heuristics for a given discrete optimization problem. While using the low-level heuristics at each stage of the search operation, the hyper-heuristics can either choose an existing heuristic or generate a new heuristic from the components of the already existing heuristics and then using the chosen or newly generated heuristic to create a new solution. The hyper-heuristics that select a heuristic from the available low-level heuristics are called selective hyper-heuristics and those that generate a new heuristic at each step of the search process are called generative hyper-heuristics. Within the selective hyper-heuristics, there are several selection mechanisms available, out of which we have used two of them, namely random selection and greedy selection. In random selection mechanism, one of the low-level heuristics is chosen randomly, whereas in greedy selection mechanism, all the low level heuristics are used to create new solutions and best solution among these new solutions is considered for further processing. Figure 1.5 shows the framework for hyper-heuristics that has the two main components. The first component is the domain-independent high level strategy and the second component has a repository of domain specific low-level heuristics. The domain-independent high level strategy is responsible for collecting and managing information such as the number of low-level heuristics, measuring the performance of the applied heuristics and keeping track of the selected heuristic, and also deciding whether to accept or reject a new solution. The other component is responsible for applying the domain specific low-level heuristics using the knowledge specific to the problem under consideration.

1.4 Scope of the thesis

In this thesis, we have worked on three facility location models and variants thereof, viz. anti-covering location problem (ACLP), obnoxious cooperative maximum coverage location problem (OCMCLP), reliability p-median problem (RpMP). We choose these models because these are

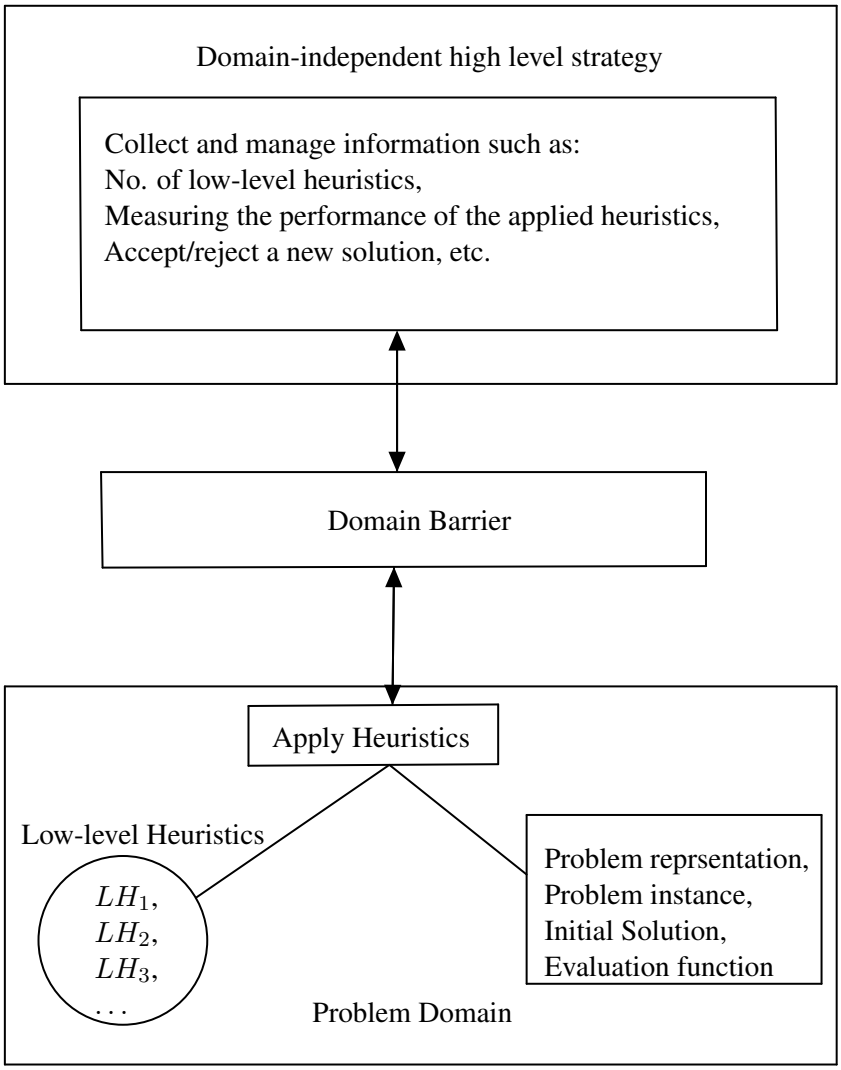


Figure 1.5: Framework of Hyper-heuristics

under-studied models despite having several real-world applications. In addition, choice of latter two models is also governed in part by their highly complex nature which pose a challenge to anyone trying to solve them. We have considered six $\mathcal{NP}$-hard facility location problems in this thesis. Out of these, first three are based on ACLP and its variants. The fourth problem

deals with location of obnoxious facilities under cooperative coverage. The last two problems are the p-median facility location problems which consider the reliability and fault tolerance issues. These problems have many practical applications in diverse fields such as locating garbage dump yards, nuclear power plants, chemical plants, telecommunication equipment, franchise outlets, liquor stores, ATMs, military defense units, DNA sequence matching, forest management, supply chain design, disaster management. As these problems are $\mathcal{NP}$-hard, applicability of exact methods is limited to small size instances only, and one has to resort to heuristic approaches to tackle instances beyond a certain maximum size. In this thesis, we have devised heuristic approaches based on genetic algorithm (GA), discrete differential evolution (DDE), and hyper-heuristics to address these problems. In addition, we have also developed some problem-specific heuristics for use within these approaches.

We have divided the thesis into seven chapters including this introductory chapter. In the following, we provide an overview of each of the remaining six chapters:

Chapter 2 deals with the anti-covering location problem (ACLP). Given a set of potential facility location sites, ACLP seeks a subset of these sites with maximum cardinality for placing the facilities in such a way that no two placed facilities are inside a specified distance of each other. ACLP has important applications in fields such as telecommunications equipment siting, locating military units, locating franchise outlets, forest management. In this chapter, we have proposed a discrete differential evolution (DDE) algorithm for this $\mathcal{NP}$-hard problem. In addition to the benchmark instances available in the literature, we have evaluated the performance of our approach on larger instances with upto 1577 nodes derived from Beasley's OR-library[1] and the standard TSPLIB[2]. Computational results show that on most of the instances, our approach performed as good as or better than the existing approaches.

Chapter 3 is concerned with two variants of the ACLP, viz. disruptive anti-covering location problem (DACLP) and weighted anti-covering location problem (WACLP). Both DACLP and WACLP are understudied facility location problems despite having several real-world applications. Given a set of potential sites for facilities, DACLP seeks to find the minimum number of facilities that can be located such that no two facilities are closer than a given distance from each other and no more facilities can be added. In competitive environments with minimum separation requirements among facilities, DACLP can be used at the minimum

[1] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/esteininfo.html
[2] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html

expanse to prevent competitors from opening more facilities in an area. WACLP considers site-dependent weights associated with all possible facility locations and it is concerned with locating a maximum weighted set of facilities in such a manner that no two facilities are within a pre-specified distance of one another.

We have extended our DDE approach for the ACLP from Chapter 2 to solve DACLP and WACLP. We have also proposed another evolutionary approach based on GA for DACLP and WACLP. Computational results show the effectiveness of our approaches in solving both the considered ACLP variants.

Chapter 4 discusses our proposed genetic algorithm (GA) based approach for solving the obnoxious cooperative maximum covering location problem (OCMCLP) on a network. In cooperative coverage models, all facilities contribute to the coverage of each demand point. A demand point is deemed to be covered if the total signal strength received by it from all the facilities is not less than a given threshold. Given a graph with the set of demand points, the set of edges between these demand points, and the non-negative real weights associated with each demand point indicating the total demand at each point, the OCMCLP is concerned with locating p obnoxious (undesirable) facilities either at the demand points or along the edges in such a manner that maximizes the uncovered demand. In all of the applications of locating obnoxious facilities, the nuisance generated by an obnoxious facility decreases over distance following some signal strength function. Facilities such as nuclear power plants, prisons, dump yards, military installations and industrial facilities causing pollution are examples of obnoxious facilities, which even though are required for the society, but produce a negative or undesirable effect. We have compared the performance of our proposed approach with two interchange heuristics available in the literature for OCMCLP. On most of the instances, our GA based approach has obtained solutions of superior quality in comparison to the existing methods.

Chapter 5 addresses the reliability p-median problem (RpMP). In the traditional p-median problem, it is assumed that once constructed, the facilities will always be available to serve the customers or demand points. But in reality, facilities may fail at times due to several reasons like natural disasters such as floods and earthquakes or due to events which are intentional like terrorist attacks and labor strikes. Sometimes facilities may fail due to unintended events like sudden power or component failures. When there are facility failures, there will be disruptions in the services provided to the customers and which force them to seek services from other functioning facilities. The reliability p-median problem minimizes both the primary transportation cost without considering the facility failures and also the cost of the expected

failure considering the facility failures. In this chapter, we have proposed a hyper-heuristic based approach with naive Bayes classifier to solve the RpMP. Ours is a multi-start greedy selection based hyper-heuristic with four low level heuristics each of which generates a feasible solution to the RpMP. We have also applied local search to further improve the fitness of the best solution. We have conducted experiments on existing benchmark instances which demonstrate that our proposed approach is able to perform better than the state-of-the-art methodologies described in the literature for RpMP.

Chapter 6 proposes two multi-start hyper-heuristic approaches for the reliable p-median problem with at-facility service (RpMF). Coming to the nomenclature of reliability p-median problem of the previous chapter, viz. Chapter 5 and reliable p-median problem of Chapter 6, we have followed the same names for both these problems as used by the respective previous authors, even though reliable p-median problem is the correct terminology grammatically. Just like RpMP, RpMF is also concerned with locating facilities where facilities may be inoperable. But, RpMF assumes that a customer doesn't have prior knowledge about the facility status until he/she reaches the facility and also applies optimized search in order to identify a facility that can provide the service by keeping track of the path that the customer has taken until he/she receives the desired service. RpMF finds its applicability in real-world examples such as bank customers withdrawing cash by visiting their nearest ATM point on regular basis which may not be servicing customers at a given time due to maintenance of the machine, people visiting petrol filling stations that have long waiting queues or shortage of petrol, patients visiting hospitals in emergency condition and are forced to seek treatment elsewhere due to long waiting times etc. We have proposed two hyper-heuristics based on greedy selection and random selection mechanisms. We have evaluated our approaches on benchmark instances and compared the results with state-of-the-art approaches available in the literature for RpMF. Our proposed approaches are able to obtain solutions of good quality in negligible execution times on majority of the instances proving the efficacy of our approaches.

Chapter 7 presents the concluding remarks of the thesis by presenting the list of contributions made in solving the aforementioned six problems. It also provides some suggestions for future research.

Chapter 2

Anti-covering location problem

2.1 Introduction

In the location science, the most common criteria for locating facilities is the interaction between a facility and the individuals which interact with that facility [78]. Apart from these facility-individual interactions, facility-facility interactions are also important, since the location of one facility may impact the location of another facility. The anti-covering location problem (ACLP) comes under the facility-facility interaction type location problems where the facilities repel each other. ACLP belongs to the class of facility location problems with minimum separation requirement between facilities.

Given a set of potential facility location sites, ACLP involves locating facilities at some of these sites in such a manner that no two facilities are within a specified distance from each other. ACLP is also referred to as r-separation problem where r is the specified distance. ACLP belongs to the class of $\mathcal{NP}$-hard problems [79, 80]. ACLP finds its importance in solving many real world applications. Some of them include but not limited to locating garbage dump yards, nuclear power plants, telecommunication equipments, franchise outlets, military defence unit location, DNA sequence matching, forest management [81].

ACLP was first defined by Moon and Chaudhry in 1984 [79] by considering the weighted variant. However, its unweighted variant received more attention than the weighted one. In the weighted variant of ACLP, each site has a positive weight associated with it as per its importance, whereas in the unweighted variant of ACLP, no weight is associated with any site. Unweighted variant can be considered as a particular case of weighted variant where all sites can be assumed to have a weight of 1. This chapter is concerned with unweighted variant only. Hereafter,

ACLP refers to its unweighted variant only. Many researchers have studied ACLP and proposed various methods to solve it [1, 79, 81, 82, 83]. In particular, Chaudhry [1] proposed an approach based on genetic algorithm. Khorjuvenkar and Singh [83] proposed a hybrid swarm intelligence approach based on Ant Colony Optimization (ACO) to solve ACLP. They have compared their approach with the approaches available in the literature and found their approach to be superior. All the approaches in the literature have solved ACLP on datasets ranging from 20 to 152 nodes. In this chapter, we applied discrete differential evolution algorithm to solve ACLP and tested it on large data sets with upto 1577 nodes. Results of the proposed approach are compared with hybrid ant colony optimization approach [83] and genetic algorithm [1].

Rest of this chapter is organized as follows: Section 2.2 formally defines the ACLP. The proposed approach for the ACLP is described in Section 2.3, and, the computational results and their analysis are presented in Section 2.4. Finally, Section 2.5 concludes the chapter by summarizing the contributions.

2.2 Formal problem definition

The ACLP can be formally defined in the following manner: Given a set $V = \{1, 2, \ldots, n\}$ of n potential facility location sites, i.e., $|V| = n$, and a distance R, so that no two facilities can be within distance R of one another. d_{uv} is the shortest distance from site $u \in V$ to site $v \in V$. The set of sites within distance R of site v is denoted by Q_v, i.e., $Q_v = \{u | u \in V \wedge d_{vu} \leq R \wedge v \neq u\}$. We call the set Q_v to be the *forbidden set* of site v. The objective of ACLP is to find a set $V' \subseteq V$ of maximum cardinality such that $Q_v \cap V' = \emptyset \; \forall v \in V'$. The constraint that no two facilities can be within distance R of one another is referred to as *separating distance constraint* subsequently. By introducing binary variables $s_v \forall v \in V$ to indicate whether site v is chosen for locating a facility ($s_v = 1$) or not ($s_v = 0$) and taking a large positive integer M, a mathematical model of ACLP, which is a modification of the formulation provided by Moon and Chaudhry [79] for the weighted variant of ACLP, is given below:

$$max \; Z = \sum_{v \in V} s_v \tag{2.1}$$

subject to the following constraints ,

$$M s_v + \sum_{u \in Q_v} s_u \leq M, \; \forall v \in V \tag{2.2}$$

$$s_v \in \{0, 1\},\ \forall v \in V \tag{2.3}$$

Here, equation 2.1 represents the objective function of the ACLP which maximizes the number of selected sites. Equation 2.2 specifies that if a facility is located at node v (i.e. $s_v = 1$), then the $Ms_v = M$, and, as a result $\sum_{u \in Q_v} s_u = 0$. So, it enforces the constraint that if a site v is part of the solution, then all the sites u within the distance R of site v, i.e., all sites belonging to Q_v can not be part of the solution. This constraint is called the neighbourhood adjacency constraint. Clearly, the value of M should be so chosen that it is larger than $\max_{v \in V}(|Q_v|)$. Constraint 2.3 enforces the binary nature of decision variables $s_v \forall v \in V$. Few alternative mathematical formulations of ACLP can be found in [81]. Throughout this chapter, we will use the term node and site interchangeably.

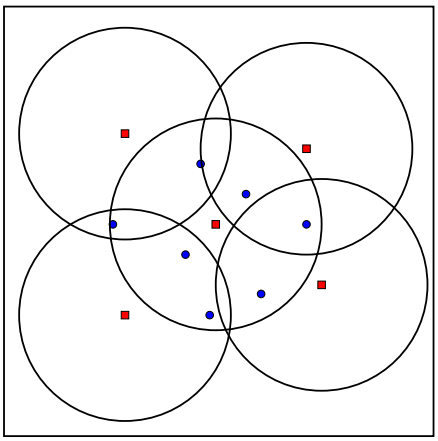


Figure 2.1: Illustration of ACLP

To illustrate ACLP, consider the example of Fig. 2.1. It contains n = 12 nodes located at different points in a plane. The coordinates of each of the nodes are as follows: A(10, 10), B(38, 10), C(30, 30), D(6, 40), E(50, 50), F(40, 40), G(70, 65), H(35, 60), I(10, 70), J(75, 20), K(70, 40) and L(55, 17). The euclidean distances from each node to all the other nodes are calculated using their respective coordinates. For the minimum separating distance R=35, the set of nodes {A, F, G, I, J} is a feasible solution, because each of these nodes are separated by minimum distance 35 from each other. The nodes belonging to this set are marked in red color, whereas other nodes are marked in blue color in the Fig. 2.1.

2.3 Proposed approach for unweighted input graph

We have developed a DDE based approach for the ACLP. The salient features of our approach are described in the following subsections.

2.3.1 Solution encoding and fitness

A bit vector of length n is used to represent a solution where a value of 1 at position t indicates a facility is located at site t. On the other hand, a value of 0 at position t indicates no facility is located at site t. We have used the objective function (equation 2.1) itself as the fitness function.

2.3.2 Initial population generation

To generate an initial solution, we need to find a subset of nodes such that no two nodes in this subset are within a distance of R from one another. In the proposed approach, each candidate solution in the initial population is generated using a randomized greedy approach. Each initial solution is generated in an iterative manner starting with an empty solution and then nodes are added to the solution one-by-one. Initially, all nodes are unmarked. During each iteration, we find k_u unmarked nodes having forbidden set of minimum cardinality (forbidden set Q_v for a node v is defined in Section 2.2). Ties are broken arbitrarily. Out of these k_u nodes, one node say v is randomly selected to be part of the solution. Now, v along with all nodes in Q_v are marked and another iteration begins. This process is repeated till no unmarked node remains. The value of k_u during each iteration is either 5 or 3. With probability P_{gen}, it is set to 5, otherwise it is set to 3.

2.3.3 DDE framework

Starting with the population of initial solutions, the discrete differential evolution approach for solving the ACLP follows an iterative process. During each iteration (referred to as generation in DE jargon), we consider each candidate solution in the population one-by-one. The solution under consideration is referred to as target solution. In the proposed approach, mutation is applied with probability P_m (i.e., mutation is not applied with probability $1 - P_m$) on the best solution found so far and the solution obtained after mutation (irrespective of whether mutation is applied or not) is called mutant or donor [71]. Followed by mutation, crossover, repair and selection procedures are applied. When all the solutions are considered, then next iteration

begins. This process is repeated for N_{iters} iterations. And the best solution found since the beginning of algorithm is returned as the final solution found by the algorithm.

2.3.4 Mutation

As part of the mutation operation, every bit of the best solution is flipped with probability P_{mut}. For every index i in the solution vector, a uniform random number r in [0,1] is generated. If r is less than P_{mut} then the corresponding bit value at index i in the best solution is flipped and copied to the mutant, otherwise the bit value from the best solution is copied unaltered to the mutant. Repair operation which is explained in the subsequent subsection (Section 2.3.6) is applied on the mutant to make it feasible and to improve its fitness.

2.3.5 Crossover

Crossover needs two solutions which act as parents to produce a new child solution. The solution obtained after mutation, viz. mutant is taken as one parent solution in the crossover and the target solution is taken as the other parent. The resulting solution after the crossover operation is called a trial solution. A simple uniform crossover operation is performed, where binary values from the mutant solution are copied to the trial solution with probability proportional to its fitness. And the binary values from the target solution are copied to the trial solution with remaining probability, which is also proportional to the target solution's fitness. Probability of copying a binary value from the mutant to the trial solution is P_{copy}=$\frac{f(mutant)}{f(mutant)+f(target\_solution)}$, and the probability of copying a binary value from the target solution to the trial solution is $1 - P_{copy}$, i.e., $\frac{f(target\_solution)}{f(mutant)+f(target\_solution)}$, where $f(X)$ is a function that computes the fitness of the solution X passed to it as argument. For every index i in the trial solution, a uniform random number, $r1$ in [0,1] is generated. If $r1$ is less than P_{copy}, then the binary value at index i from the mutant is copied to the trial solution at index i, otherwise the target solution's binary value at index i is copied to the trial solution at index i. This is repeated for all the n indices. The crossover is applied with probability P_c when mutation has already been applied. Otherwise, it is always applied. This is done to prevent the trial solution from being an exact copy of the best solution.

2.3.6 Repair

As there is no guarantee of the feasibility of the trial solution obtained after the crossover, repair operation is performed on the trial solution. In addition to transforming an infeasible solution into a feasible solution, the repair operation also tries to improve the solution. In the repair operation, first it is checked whether the trial solution is feasible or not by verifying the separating distance constraint. If the trial solution T is not feasible, then we compute $Q_v^{'} = \{u | u \in T \wedge d_{vu} \leq R \wedge v \neq u\} \forall v \in T$ and a site v with maximum $|Q_v^{'}|$ value is removed from T by setting the corresponding bit in the solution to 0. This entire process is repeated till the trial solution becomes feasible.

Once the trial solution is made feasible, we make an attempt to increase its fitness if the latest fitness is within 20% of the best solution's fitness. To maintain a balance between solution quality and the execution time, we arrived at this 20% after large number of experiments. To improve the fitness, we will compute the set S_{rem} of all those remaining sites which can still be added to the solution without violating the separating distance constraint. Then, we compute $Q_u^{''} = Q_u \cap S_{rem}\ \forall u \in S_{rem}$. All those sites v in S_{rem} such that $Q_v^{''} = \emptyset$ are added immediately to the solution by setting the corresponding bits in the solution to 1. If there is no site $v \in S_{rem}$ with $Q_v^{''} = \emptyset$, then a site $v \in S_{rem}$ which has the minimum value of $|Q_v^{''}|$ (ties are broken arbitrarily) is made part of the solution by marking the corresponding bit in the solution as 1. The set S_{rem} is updated to reflect the change in configuration of the solution. This process is repeated till S_{rem} becomes empty.

2.3.7 Selection

After the repair operation, the fitness of trial solution is compared with the fitness of the target solution. If the trial solution has higher fitness than the target solution, then it replaces the target solution in the population, otherwise the target solution remains in the population for the next generation and the trial solution is discarded.

Algorithm 3 provides the pseudo-code for our discrete differential evolution approach where $u01$ is a uniform variate in [0, 1]. *Mutation*, *Crossover*, *Repair* and f are four functions that perform mutation (Section 2.3.4), crossover (Section 2.3.5), repair (Section 2.3.6) and fitness computation (Section 2.3.1) operations respectively.

Algorithm 3: Discrete differential evolution algorithm for ACLP

```
Generate initial population;
best_solution ← best solution in initial population;
iter ← 0;
while iter < N_iters do
    foreach target_solution ∈ population do
        if u01 ≤ P_m then
            mutant ← Mutation(best_solution);
            no_mutation ← 0;
        else
            mutant ← best_solution;
            no_mutation ← 1;
        if (u01 ≤ P_c) or (no_mutation = 1) then
            trial_solution ← Crossover(mutant, target_solution);
        else
            trial_solution ← mutant;
        trial_solution ← Repair(trial_solution);
        if f(trial_solution) ≥ f(target_solution) then
            target_solution ← trial_solution;
            if f(trial_solution) ≥ f(best_solution) then
                best_solution ← trial_solution;
    iter ← iter + 1;
return best_solution;
```

2.4 Experimental results

We have implemented our DDE approach in C. In all our experiments with DDE, we have used population size (NP) = 50, P_{gen} = 0.5, P_m = 0.9, P_c = 0.9, and P_{mut} = 0.02. All these parameter values are chosen empirically.

We have used three datasets in our experiments. The first dataset was used in [1] to test the performance of genetic algorithm (GA). Later, it was used in [83] to test the performance of proposed ant colony optimization (ACO) approach. This dataset consists of 41 instances with number of nodes either 20 or 30 or 55 and different values of R. Additionally, we have used two more datasets containing larger instances derived from Beasley's OR-library[1] and the standard TSPLIB[2]. The datasets derived from OR library contain 40 ACLP instances and have number of nodes from the set {50, 100, 250, 500 and 1000} and R from the set {5, 10, 25, 50}. The datasets derived from TSPLIB also contain 40 ACLP instances and have number of nodes varying from 51 to 1577. For the TSPLIB datasets, we have taken the R values as

[1] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/esteininfo.html
[2] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html

Table 2.1: Comparison of objective values of GA, ACO and DDE based solutions for datasets used in [1]

Dataset	R	O_{known}	O_{GA}	O_{ACO}	O_{DDE}
20 nodes	50	12	12	12	12
	100	12	12	12	12
	150	9	9	9	10
	200	9	9	9	9
	250	8	8	8	8
	300	7	7	7	7
	350	6	6	6	6
	400	6	6	6	6
	450	6	6	6	6
	500	5	5	5	5
	550	5	5	5	5
	600	5	5	5	5
	650	5	5	5	5
	700	3	3	3	3
	750	3	3	3	3
	800	3	3	3	3
	850	3	3	3	3
	900	3	3	3	3
	950	2	2	2	2
	1000	1	1	1	1
30 nodes	35	24	24	24	24
	50	21	21	21	21
	100	10	10	10	10
	150	6	6	6	6
	200	4	4	4	4
	300	3	3	3	3
	400	2	2	2	2
55 nodes	5	36	36	36	36
	6	30	30	30	30
	7	25	24	25	25
	8	21	20	21	21
	9	18	17	18	18
	10	16	15	16	16
	12	13	12	13	13
	14	11	10	11	11
	15	9	9	9	9
	20	7	6	7	7
	23	6	5	6	6
	27	5	4	5	5
	31	4	4	4	4
	34	3	3	3	3

5%, 10%, 25% and 50% of the $\frac{X_{max}-X_{min}+Y_{max}-Y_{min}}{2}$ where X_{max}, X_{min}, Y_{max} and Y_{min} are respectively the maximum X value, minimum X value, maximum Y value and minimum Y value over all points in the corresponding base TSPLIB instance. For the first dataset, we have compared our DDE approach with ACO and GA, whereas for remaining two datasets, which are created by us, we have compared our DDE approach with ACO only. The reason for using only ACO for comparison for latter two datasets is availability of the source code of ACO for execution on these new datasets. Further, this is fair also as superiority of ACO over other approaches in the literature on first dataset has been shown already in [83].

We have executed our DDE approach and the ACO approach for the same amount of time on a Linux based 3.40 GHz Core-i5-7500 system with 8 GB RAM. On all the instances with number of nodes less than or equal to 100, these two approaches are run for 1 second, on all the instances with number of nodes greater than 100 and upto 500, these two approaches are executed for 2 seconds, and on all the instances with more than 500 nodes, these two approaches are executed for 5 seconds. These two approaches are executed 10 independent times on each instance like GA. For first dataset, ACO and GA found the same solution in all the 10 runs on all the 41 instances.

Table 2.1 presents the results of our DDE approach(column O_{DDE}) on 41 instances used in [1] and compares them with GA (column O_{GA}), ACO (column O_{ACO}) and best known solutions (column O_{known}). Comparison among DDE, ACO and GA is done in terms of best solution obtained over 10 runs to ensure conformity with the results reported in [1]. Data for GA is taken from [1]. Both DDE and ACO found best known solutions on all 41 instances, whereas GA fails to find the same on some instances.

Table 2.2 and Table 2.3 present the results of DDE and ACO approaches on OR library and TSPLIB datasets respectively. Results are reported in terms of best and average solution quality obtained over 10 runs on each instance. In these tables, best results are in bold font. Table 2.4 is the summary table listing the number of instances on which DDE obtained worse ($<$), same ($=$) and better ($>$) solutions in comparison to ACO. This is done for each of the two datasets and overall. These three tables clearly show the superiority of our DDE approach over ACO. On most of the instances our approach performed as good as or better than ACO approach. Only on few instances, our approach performed worse than the ACO approach.

We have also done statistical significance analysis of our approach in comparison to ACO based approach. We have performed Wilcoxon Signed-Ranks test [84] for N = 80 instances of the OR library and TSPLIB datasets together, with a significance level α = 0.01. As shown in Table 2.5 the z value obtained, -3.447, is less than the critical value of z, z_c= -2.33 for a two-tailed test. This proves that the improvement achieved with our approach is significant and it is due to the algorithmic merit rather than random fluctuations.

Table 2.2: Comparison of objective values of ACO and DDE based solutions for OR library datasets

		ACO Solution		DDE Solution	
Dataset	R	Best	Average	Best	Average
OR_50.1	5	**43**	**43.00**	**43**	**43.00**
	10	**31**	**31.00**	**31**	**31.00**
	25	**12**	**12.00**	**12**	**12.00**
	50	**6**	**6.00**	**6**	**6.00**
OR_50.2	5	**40**	**40.00**	**40**	**40.00**
	10	**32**	**32.00**	**32**	**32.00**
	25	**12**	**12.00**	**12**	**12.00**
	50	**6**	**6.00**	**6**	**6.00**
OR_100.1	5	**67**	**67.00**	**67**	**67.00**
	10	**39**	**39.00**	**39**	**39.00**
	25	**13**	**13.00**	**13**	**13.00**
	50	**6**	**6.00**	**6**	**6.00**
OR_100.2	5	**71**	**71.00**	**71**	**71.00**
	10	**43**	**43.00**	**43**	**43.00**
	25	**15**	**15.00**	**15**	**15.00**
	50	**5**	**5.00**	**5**	**5.00**
OR_250.1	5	**131**	**131.00**	**131**	**131.00**
	10	**62**	**61.20**	61	60.50
	25	**17**	**17.00**	**17**	**17.00**
	50	**6**	**6.00**	**6**	**6.00**
OR_250.2	5	**127**	**127.00**	**127**	**127.00**
	10	**62**	**62.00**	**62**	**62.00**
	25	**17**	**17.00**	**17**	**17.00**
	50	**6**	**6.00**	**6**	**6.00**
OR_500.1	5	**179**	178.90	**179**	**179.00**
	10	**74**	73.00	**74**	**73.20**
	25	**19**	**19.00**	**19**	18.30
	50	**7**	6.90	**7**	**7.00**
OR_500.2	5	178	177.60	**179**	**178.40**
	10	**74**	**73.10**	**74**	72.40
	25	**18**	**18.00**	**18**	**18.00**
	50	**6**	**6.00**	**6**	**6.00**
OR_1000.1	5	217	215.70	**227**	**225.40**
	10	**83**	**82.60**	**83**	81.90
	25	**20**	19.20	**20**	**19.60**
	50	**7**	**7.00**	**7**	**7.00**
OR_1000.2	5	212	211.10	**220**	**217.40**
	10	81	**80.50**	**82**	80.20
	25	**20**	19.10	**20**	**19.70**
	50	**7**	**7.00**	**7**	**7.00**

Table 2.3: Comparison of objective values of ACO and DDE based solutions for TSPLIB datasets

		ACO Solution		DDE Solution	
Dataset	R	Best	Average	Best	Average
eil51	3	**50**	**50.00**	**50**	**50.00**
	6	**39**	**39.00**	**39**	**39.00**
	15	**14**	**14.00**	**14**	**14.00**
	30	**6**	**6.00**	**6**	**6.00**
rat99	8	**85**	**85.00**	**85**	**85.00**
	15	**47**	**47.00**	**47**	**47.00**
	38	**14**	**14.00**	**14**	**14.00**
	75	**6**	**6.00**	**6**	**6.00**
rat195	11	**125**	**125.00**	**125**	**125.00**
	21	59	59.00	**60**	**59.10**
	52	**16**	**16.00**	**16**	**16.00**
	104	**6**	**6.00**	**6**	**6.00**
pr299	223	**118**	**118.00**	**118**	**118.00**
	446	57	56.90	**58**	**57.60**
	1114	**16**	**16.00**	**16**	**16.00**
	2228	**6**	**6.00**	**6**	**6.00**
d493	170	89	89.00	**90**	**89.10**
	340	**38**	**37.30**	37	37.00
	851	**10**	**10.00**	**10**	**10.00**
	1700	**4**	**4.00**	**4**	**4.00**
u724	111	211	211.00	**212**	**211.10**
	222	78	76.60	**79**	**77.20**
	555	**18**	**18.00**	**18**	**18.00**
	1110	**6**	**6.00**	**6**	**6.00**
pr1002	650	**200**	195.60	198	**197.00**
	1300	**76**	**76.00**	**76**	74.70
	3250	**18**	**18.00**	**18**	**18.00**
	6500	**6**	**6.00**	**6**	**6.00**
pcb1173	120	254	250.90	**258**	**256.60**
	240	**86**	**85.30**	85	83.80
	600	**19**	**19.00**	**19**	**19.00**
	1200	**6**	**6.00**	**6**	**6.00**
d1291	176	**133**	**132.80**	**133**	**132.80**
	352	**50**	**49.90**	**50**	49.20
	879	13	13.00	**14**	**13.20**
	1760	**5**	**5.00**	**5**	**5.00**
fl1577	91	98	95.10	**99**	**98.60**
	182	**50**	49.10	**50**	**50.00**
	456	**15**	**15.00**	**15**	**15.00**
	910	**6**	**6.00**	**6**	**6.00**

Table 2.4: Comparison Summary: Number of instances on which DDE obtained worse ($<$), same ($=$) and better ($>$) solutions in comparison to ACO

	Best			Average		
Dataset name	$<$	$=$	$>$	$<$	$=$	$>$
OR library	1	35	4	5	27	8
TSPLIB	3	29	8	4	26	10
Overall	4	64	12	9	53	18

Table 2.5: Wilcoxon Signed-Ranks test of our approach with ACO

N	W^+	W^-	z_c	z
80	143	637	-2.33	-3.447

2.5 Conclusions

In this chapter we have proposed a population based solution, viz. a discrete differential evolution algorithm for the ACLP. We have evaluated and compared our approach with the state-of-the approaches on the benchmark instances used in [1, 83]. We have also generated new datasets containing larger number of nodes, and compared our approach with the ACO approach proposed in [83] on these new datasets. Computational results show that on most of the instances, our approach performed as good as or better than the ACO approach. Only for few datasets, the ACO approach performed better than our approach.

Chapter 3

Two ACLP variants

3.1 Introduction

In this chapter, we discuss two variants of the anti-covering location problem, viz. disruptive anti-covering location problem (DACLP) and weighted anti-covering location problem (WACLP). Both DACLP and WACLP are understudied facility location problems and are related to the anti-covering location problem (ACLP) discussed in the previous chapter. DACLP was introduced in the last decade by Niblett and Church [85] while WACLP was introduced by Moon and Chaudhry [79] for the first time in 1984. Over the past many years there have been several methods devised for solving ACLP [1, 79, 81, 82, 83]. However, no method exists in the literature to solve DACLP other than the ILP proposed by Niblett and Church [85] while introducing DACLP. Similarly, for WACLP, only the ILP proposed by Moon and Chaudhry [79] while introducing WACLP and the four greedy heuristic approaches [86] exist in the literature. So these two problems are understudied problems.

Motivated by the understudied nature of DACLP and WACLP as explained in the previous paragraph, and the very fact that different evolutionary algorithms have already been used successfully to solve innumerable combinatorial optimization problems (e.g.,[87, 88, 89, 90, 91, 92, 93]), we have proposed two evolutionary approaches to solve DACLP and WACLP. As our first approach, we have extended our discrete differential evolution (DDE) algorithm based approach for the ACLP to both the considered problems, and, our second approach is based on genetic algorithm (GA). Though differential evolution and genetic algorithm both belong to the broad class of evolutionary algorithms and make use of crossover and mutation, the solution encoding, crossover and mutation used by our two approaches are entirely different.

We have evaluated the performance of our approaches on the 80 ACLP instances with upto 1577 nodes which are introduced in Chapter 2. We have used the instances from Chapter 2 in the same form in the case of DACLP and modified them to have node weights in the case of WACLP. For DACLP, we have reported the results of the two proposed approaches and presented the comparative analysis. In the case of WACLP, the results of the proposed approaches are compared with the four greedy heuristics proposed in [86]. Computational results show the superiority of our approaches in comparison to these greedy heuristics.

We have decided to devote a single chapter for these two ACLP variants instead of two separate chapters because of the similar nature of approaches that we have developed for them.

Remainder of this chapter is organized into three sections. Section 3.2 is devoted to DACLP, whereas Section 3.3 is devoted to WACLP. These two sections have several subsections, each presenting the details of the proposed approaches. Finally, Section 3.4 wraps up the chapter by listing the contributions made.

3.2 Disruptive ACLP

The disruptive anti-covering location problem (DACLP) comes under the facility-facility interaction type location problems where no two facilities can be located within a distance of R from one another. In the DACLP jargon [85], a proper solution is defined as the one in which all non-facility sites are within the separating distance R from one or more of the selected facilities. Obviously, no more facilities can be added to a proper solution. So, DACLP is concerned with finding the minimum number of facilities that can be located on a subset of sites while giving a proper solution. DACLP is derived from the more commonly known anti-covering location problem (ACLP) [79], which is concerned with finding a subset of facilities of maximum cardinality which forms a proper solution. The previous chapter discusses ACLP in detail. The disruptive anti-covering location problem is so named as it prevents the "best or maximal" packing solution of the anti-covering location problem from occurring. Node and site have been used synonymously throughout this chapter.

Niblett and Church [85] introduced DACLP for the first time in 2015 and proposed a model based on integer linear programming (ILP) for solving this problem. DACLP is an $\mathcal{NP}$-hard problem [85]. There are many real world applications where DACLP can be used for finding the minimum number of facilities that can be located with the minimum separating distance requirement between each pair of facilities. In competitive environments where there

are minimum separation requirements among facilities, DACLP can be used at the minimum expanse to prevent competitors from opening more facilities in an area. For example, if there is a minimum separation requirement between any two liquor stores in a city then opening of liquor stores by a company as per DACLP solution for this city at minimum cost will forbid competitors from opening any more stores in that city [85, 94]. Also in applications like analyzing policies impacting potential sex offenders' residence locations [95], and carrying capacity of a population of Sandhill Cranes [85] etc., DACLP generates important and informative solutions. Apart from these, in all the applications of ACLP that involve independent decision making entities, the solutions found through DACLP are significant in the decision making and policy analysis.

This section is divided into various sub-sections in the following manner: Section 3.2.1 gives the formal definition of DACLP. Section 3.2.2 presents the proposed DDE approach, whereas Section 3.2.3 describes the proposed GA approach for the DACLP. The results of the conducted experiments along with their analysis are presented in 3.2.4.

3.2.1 Problem definition

Considering a set V of n potential sites where facilities can be located, i.e., $V = \{1, 2, \ldots, n\}$ ($|V| = n$), and R is the minimum separating distance such that no two facilities are permitted within distance R from one another, the disruptive anti-covering location problem can be formally defined as follows: For each site $u \in V$, the shortest distance between site u and site $v \in V$ is given by d_{uv}. Q_u represents the forbidden set of site $u \in V$, i.e., $Q_u = \{v | v \in V \wedge d_{uv} \leq R \wedge u \neq v\}$. A solution with set $S \subseteq V$ of facilities is called proper in case facilities are located in such a manner that no two facilities are located within distance R from one another and all non-facility sites are within a distance R from one or more facilities, i.e.,$Q_u \cap S = \emptyset$ $\forall u \in S$ and $(\cup_{u \in S} Q_u) \cap \{v\} \neq \emptyset \, \forall v \in (V \setminus S)$. DACLP seeks a proper solution with minimum number of facilities. Considering binary variables $x_v \forall v \in V$ that have value 1 if a facility is located at site v ($x_v = 1$) and value 0 when no facility is located at site v ($x_v = 0$) and Y as a large positive integer, Niblett and Church [85] formulated the following mathematical model of DACLP:

$$min\ Z = \sum_{u \in V} x_u \tag{3.1}$$

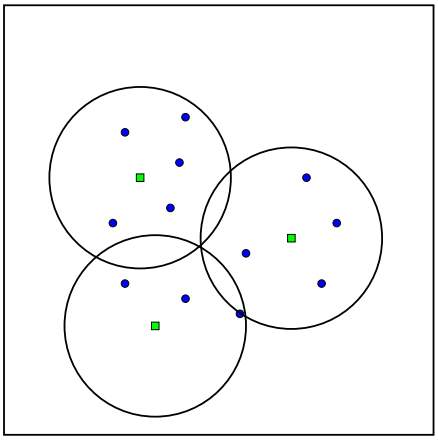


Figure 3.1: DACLP illustration

subject to:

$$Yx_u + \sum_{v \in Q_u} x_v \leq Y, \ \forall u \in V \tag{3.2}$$

$$x_u + \sum_{v \in Q_u} x_v \geq 1, \ \forall u \in V \tag{3.3}$$

$$x_u \in \{0, 1\}, \ \forall u \in V \tag{3.4}$$

Here, equation 3.1 minimizes the number of sites where facilities are located and gives the DACLP's objective function. According to equation 3.2 if a site u is selected for locating a facility (i.e. $x_u = 1$), then the $Yx_u = Y$, and, which causes the summation $\sum_{v \in Q_u} x_v$ to be 0, i.e., it enforces the constraint that no two facilities can be within distance R from one another. Constraint 3.3 enforces that either site u is selected to locate a facility or a site $v \in Q_u$ which is within R distance from u is selected to locate a facility. Together equations 3.2 and 3.3 make sure that the solution is proper. Constraint 3.4 restricts the variables $x_u \forall u \in V$ to binary values.

When compared with the mathematical formulation of ACLP (Section 2.2) from previous chapter, the objective function of DACLP is a minimization function as given in Equation 3.1. Apart from the objective function, DACLP has an additional constraint given in the Equation 3.3 that makes sure that no site is left uncovered.

Consider the Fig. 3.1 which illustrates DACLP with an example. There are a total of 15 nodes having the following coordinates: A(10, 20), B(30, 15), C(20, 6), D(48, 10), E(30, 75), F(6, 40), G(50, 30), H(65, 35), I(80, 40), J(70, 55), K(28, 60), L(25, 45), M(10, 70), N(15, 55)

and O(75, 20). Considering R = 30, where all the facilities are separated from each other by a distance of more than R, the subset {C, H, N} forms a feasible, proper solution satisfying the separating distance constraint. In the Fig. 3.1 we have depicted the nodes selected for facilities in green color and nodes not selected for facilities are marked in blue color.

3.2.2 DDE approach for DACLP

We have extended our discrete differential evolution (DDE) based approach for ACLP presented in the previous chapter to the disruptive ACLP by making the required changes in the approach. Subsequent subsections describe the salient features of our DDE approach for the DACLP.

3.2.2.1 Solution representation and fitness

We have used the bit vector representation to represent a solution just like in the DDE approach for ACLP from the previous chapter (Section 2.3.1). For the fitness function, we have used the the objective function of DACLP which is given in equation 3.1.

3.2.2.2 Generating initial population

To generate each member of the initial population, we have used a semi-greedy method which is an extension of the method used in Chapter 2 (Section 2.3.2). The following are the two differences for the initial solution generation in DDE approach for DACLP as compared to the same in DDE approach for ACLP. Since DACLP is minimization problem, nodes with highest values of $|Q_u|$ are given preference whereas in ACLP nodes with least values of $|Q_u|$ are given preference as ACLP is a maximization problem. In DACLP, we always determine 3 unmarked sites with highest values of $|Q_u|$ and out of these 3 sites, we randomly select a site u to be part of the solution. Whereas in ACLP, k_u unmarked nodes with least values of $|Q_u|$ are selected where in each iteration k_u is set to 5 with probability P_{gen}, otherwise it is set to 3. Except for these aforementioned differences, the method of initial solution generation in the DDE approach for DACLP is the same as in the DDE for ACLP. In this manner, a population of total POP_{cnt} candidate solutions are generated.

3.2.2.3 DDE framework

The same DDE framework used for the ACLP in the previous chapter(Section 2.3.3) has been used for the DACLP also with the modification of using minimization in place of maximization.

We performed mutation on the global best solution just like in the previous chapter. As part of the uniform crossover performed on mutant and target solution, we copy bit values from the mutant to the trial solution with probability p_{copy}=$\frac{f(target_{sol})}{f(mutant)+f(target_{sol})}$ or else bit values are copied from the target solution. This is different from the policy used in the crossover used in the previous chapter (Section 2.3.5) to account for the change in nature of the objective of the problem from maximization to minimization. After the crossover, repair operation is performed on the trial solution which is explained below. Apart from these just mentioned differences, the DDE framework for DACLP is the same as in the previous chapter.

3.2.2.4 Repair

In the repair operation, if the solution obtained through mutation/crossover is infeasible because the separating distance constraint among facility sites is not satisfied, we eliminate some facilities to make it a feasible solution. A randomized approach is followed for eliminating facilities from the given solution. We randomly select a site u which is part of the current solution and mark all the sites which are in its forbidden set Q_u as not being part of the solution. We repeat this step until the trial solution is made feasible.

After the trial solution becomes feasible, we check whether it is a proper solution or not. If it is not proper, then to add new facilities, we find the set X_{rem} which contains the sites that can be part of the solution without making the solution infeasible. Then, a new set $Q_v^{'} = Q_v \cap X_{rem}$ $\forall v \in X_{rem}$ is computed. After that a site $u \in X_{rem}$ with the highest cardinality of $Q_u^{'}$ is added to the solution. Then we update the sets X_{rem}, $Q_u^{'}$ $\forall u \in X_{rem}$ according to the latest changes in the solution. The repair procedure stops once X_{rem} becomes empty.

The pseudo-code for our DDE approach for DACLP is given in Algorithm 4. The mutation, crossover, repair (Section 3.2.2.4) and fitness computation (Section 3.2.2.1) operations are performed by the four functions *Mutation*, *Crossover*, *Repair* and *fitness* respectively. $r01$ is a uniform variate in [0, 1].

3.2.3 GA approach for DACLP

A steady-state genetic algorithm [66] is the other evolutionary approach that we have proposed for the DACLP. In the remainder of this section, we refer to this approach as GA. The following subsections present the important features of the proposed GA approach.

Algorithm 4: DDE algorithm for DACLP

```
Generate initial population;
best_sol ← best solution from the initial population;
while (termination condition remains unsatisfied) do
    foreach (target_sol ∈ population) do
        if (r01 ≤ p_m) then
            no_mutation ← 0;
            mutant_sol ← Mutation(best_sol);
        else
            mutant_sol ← best_sol;
            no_mutation ← 1;
        if ((no_mutation = 1) or (r01 ≤ p_c)) then
            trial_sol ← Crossover(target_sol, mutant_sol);
        else
            trial_sol ← mutant_sol;
        trial_sol ← Repair(trial_sol);
        if fitness(trial_sol) ≤ fitness(target_sol) then
            target_sol ← trial_sol;
            if fitness(trial_sol) ≤ fitness(best_sol) then
                best_sol ← trial_sol;
return best_sol;
```

3.2.3.1 Solution representation and fitness

A solution in our GA approach represents the sites selected for locating facilities as an ordered list. It is an efficient representation compared to bit-vector as it consumes less memory to store a solution and also requires less computation time in the overall operations. Even though ordered list causes sorting overhead, such an encoding allows the efficient implementation of variation operators like crossover and mutation as explained in corresponding subsections. The chromosome length in this representation is not fixed as in the bit-vector representation, but the variation operators are designed accordingly.

For our GA approach also, we have taken the objective function as the fitness function in the same way as in our proposed DDE approach.

3.2.3.2 Initial solution generation

To generate the initial solutions, we have used a method which is a combination of a completely random method and a semi-greedy method. In this method, to begin with, we consider all the n sites as unvisited and start with an empty set for the solution, then we follow an iterative procedure. In each iteration, with probability ρ_{rd}, we randomly chose an unvisited site u, and make it part of the solution. As part of the semi-greedy method, 5 unvisited sites with highest value of $|Q_v|$ are determined and one of these 5 sites is randomly selected and is added to

the solution. Considering the newly added site as u, we mark the site u and every site in its forbidden set as visited, and proceed to the next iteration. Till there are no unvisited sites remaining, this process is repeated. Based on the number of unvisited sites, following are some exceptions to the aforementioned rules of selecting a site in an iteration. If there is only a single unvisited site, then it is added directly. If there are two sites which are unvisited, then the one having highest cardinality of the forbidden set is selected. On the other hand, if there are more than 2 and less than 6 unvisited sites, we randomly choose one site from among the unvisited sites. After generating a complete solution, we make it an ordered list by sorting.

3.2.3.3 Selection

The two parent solutions for crossover and a single parent for mutation are chosen using probabilistic binary tournament selection in which parameter ρ_{pbt} gives the probability based on which the fitter of the two randomly chosen solutions from the population is selected to be a parent.

3.2.3.4 Crossover

As part of the crossover, we first determine the intersection set of sites present in two parents. As solutions are represented as ordered lists, time taken to find the intersection of the parent solutions S_1 and S_2 is only $\mathcal{O}(min(|S_1|, |S_2|))$ instead of $\mathcal{O}(|S_1|.|S_2|)$. We copy the sites from the intersection set to the child, as the sites occurring in both the parents have a higher chance of being part of several good solutions. After this, in a similar method followed in generating initial solutions, remaining sites are selected to be part of the child solution one at a time, but value of ρ_{rd} can vary. We set the value of ρ_{rd} to zero with probability ρ_{add}, otherwise the same ρ_{rd} value as in initial solution generation is used.

3.2.3.5 Mutation

For every site present in the parent solution, we generated a uniform random number $u01 \in [0, 1]$. Only if $u01$ is less than ρ_m, the corresponding site is copied to the mutant, otherwise it is not copied to the mutant. After repeating this for all the sites in the parent solution, we have followed the same method as in crossover to add other sites to the mutant.

In our GA approach, we have utilized crossover and mutation in a mutually exclusive manner. Crossover is utilized with probability ρ_c, and with the remaining probability of $1 - \rho_c$ mutation

Algorithm 5: GA for DACLP

```
Construct ps initial solutions X_1, X_2, ..., X_ps;
X_best ← best solution among ps initial solutions;
while (termination condition remains unsatisfied) do
    if (u01 < ρ_c) then
        S_1 ← BTS(X_1, ..., X_ps);
        repeat
            S_2 ← BTS(X_1, ..., X_ps);
        until (S_1 ≠ S_2);
        X_C ← Cross(S_1, S_2);
    else
        S_1 ← BTS(X_1, ..., X_ps);
        X_C ← Mutate(S_1);
    X_C ← Localsearch(X_C);
    Include X_C in the population as per replacement policy;
    if (X_C is better than X_best) then
        X_best ← X_C;
return X_best;
```

is utilized. The reason being, as part of crossover operator, we retain the common sites which are in both the parent solutions so as to generate even better child solutions using these common sites. If mutation is applied after the crossover, some of these sites which are common in both the parents will be deleted.

3.2.3.6 Population replacement model

We have used a steady-state population replacement model in our GA approach. In this model, every generation produces only a single child solution. The child solution is discarded if it is found to be same as any of the existing members of the population. Otherwise, it replaces the member with the worst fitness if its fitness is better than that of the worst fitness member.

3.2.3.7 Local search

After crossover/mutation, to further minimize the child solution fitness, we have performed a two-one exchange operation as part of the local search. In this local search, we replace a pair of sites in the solution with a single site only if the resulting solution is proper and feasible.

Algorithm 5 gives the pseudo-code for the proposed GA where the probabilistic binary tournament selection method (Section 3.2.3.3), crossover operator (Section 3.2.3.4), mutation operator (Section 3.2.3.5) and local search (Section 3.2.3.7) are carried out by four functions $BTS()$, $Cross()$, $Mutate()$ and $Localsearch()$ respectively. Further, $u01$ is a uniform random variate in [0, 1] and ps is the population size.

Table 3.1: Parameters for DDE and GA

DDE Parameters		GA Parameters	
Parameter	Value	Parameter	Value
POP_{cnt}	250	ps	250
p_m	0.9	ρ_{rd}	0.75
p_c	0.9	ρ_{bts}	0.8
p_{mut}	0.02	ρ_c	0.5
		ρ_m	0.75
		ρ_{add}	0.9

3.2.4 Experimental results

Both of our proposed approaches, viz. DDE and GA have been implemented in C. Table 3.1 lists the different parameters involved and their corresponding values for DDE and GA based approaches both. The respective parameter values of both the proposed approaches are chosen empirically. We have run both our approaches on a Linux system with 8 GB RAM and 3.40 GHz Core-i5-7500 processor. For each test instance, we have performed 10 independent runs of DDE and GA. We have fixed the same maximum execution time for each run of both the proposed approaches. We have executed both DDE and GA for 10 seconds on those instances having number of nodes upto 100. On those instances having more than 100 and upto 500 nodes, we have run both the proposed approaches for 60 seconds, and on the remaining instances having number of nodes greater than 500, we have run the the two approaches for 100 seconds.

We have tested our approaches on two different types of datasets that are derived from Beasley's OR-Library[1] and the TSPLIB[2] which we have first introduced in the previous chapter. There are 40 instances in the dataset derived from OR library with the number of nodes in the range of 50 to 1000 and R value in the range of 5 to 50. Similarly, there are 40 instances derived from TSPLIB with the number of nodes in the range of 51 to 1577 and R values of TSPLIB instances are considered as mentioned in the previous chapter.

Table 3.2 presents the results obtained by our proposed approaches DDE and GA on OR-Library instances, while Table 3.3 presents the results obtained by our proposed approaches on TSPLIB instances. In both the tables 3.2 and 3.3, the 1st column, *Instance*, is the dataset name. Column two, R, gives the distance within which no two facilities can be located. The least and average solution values of the DDE method over 10 independent runs are given in columns 3, 4 and the least and average solution values of the GA method over 10 independent runs are given in columns 5, 6 respectively. The least objective value across all techniques is highlighted in

[1] `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/esteininfo.html`
[2] `http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html`

Table 3.2: Results on OR-Library dataset for DDE and GA

		DDE		GA	
Instance	R	Least	Average	Least	Average
OR_50.1	5	**42**	**42.00**	**42**	**42.00**
	10	**26**	**26.00**	**26**	**26.00**
	25	**8**	**8.00**	**8**	**8.00**
	50	**3**	**3.00**	**3**	**3.00**
OR_50.2	5	**39**	**39.00**	**39**	**39.00**
	10	**26**	**26.00**	**26**	**26.00**
	25	**7**	**7.00**	**7**	**7.00**
	50	**3**	**3.00**	**3**	**3.00**
OR_100.1	5	**60**	**60.00**	**60**	**60.00**
	10	**29**	**29.00**	**29**	**29.00**
	25	**7**	**7.00**	**7**	**7.00**
	50	**3**	**3.00**	**3**	**3.00**
OR_100.2	5	**64**	**64.00**	**64**	**64.00**
	10	**31**	**31.00**	**31**	**31.00**
	25	**7**	**7.00**	**7**	**7.00**
	50	**3**	**3.00**	**3**	**3.00**
OR_250.1	5	**104**	**104.00**	**104**	**104.00**
	10	**34**	**34.90**	35	35.40
	25	**8**	**8.00**	**8**	**8.00**
	50	**3**	**3.00**	**3**	**3.00**
OR_250.2	5	**93**	**93.00**	**93**	**93.00**
	10	**33**	**33.00**	**33**	33.20
	25	**7**	**7.00**	**7**	7.90
	50	**3**	**3.00**	**3**	**3.00**
OR_500.1	5	**114**	**114.30**	117	117.90
	10	**35**	**35.50**	37	38.00
	25	**8**	**8.00**	**8**	8.40
	50	**3**	**3.00**	**3**	**3.00**
OR_500.2	5	**110**	**110.50**	113	114.00
	10	**35**	**35.20**	37	37.40
	25	**8**	**8.00**	**8**	**8.00**
	50	**3**	**3.00**	**3**	**3.00**
OR_1000.1	5	**125**	**127.20**	137	140.40
	10	**38**	**39.70**	41	42.40
	25	**8**	**8.00**	**8**	8.90
	50	**3**	**3.00**	**3**	**3.00**
OR_1000.2	5	**123**	**124.70**	133	134.70
	10	**38**	**39.60**	42	43.20
	25	**8**	**8.00**	**8**	8.90
	50	**3**	**3.00**	**3**	**3.00**

bold for easy identification. Table 3.4 provides the summary of results in terms of number of instances on which DDE obtained better solution (<), same solution (=) and worse solution (>) when compared with GA. This summary is provided for the least objective values and average objective values both.

On the OR-Library dataset, for the least objective value over 10 independent runs, out of the 40 instances DDE produced the smaller objective values on 9 instances and the same objective value as GA on 31 instances. For the average objective values of 10 independent runs on the OR library dataset, DDE produced the smaller average values as compared to GA on 16 instances

Table 3.3: Results on TSPLIB dataset for DDE and GA

		DDE		GA	
Instance	R	Least	Average	Least	Average
eil51	3	**50**	**50.00**	**50**	**50.00**
	6	**37**	**37.00**	**37**	**37.00**
	15	**8**	**8.00**	**8**	**8.00**
	30	**3**	**3.00**	**3**	**3.00**
rat99	8	**82**	**82.00**	**82**	**82.00**
	15	**31**	**31.00**	**31**	**31.00**
	38	**7**	**7.00**	**7**	**7.00**
	75	**2**	**2.00**	**2**	**2.00**
rat195	11	**106**	**106.00**	**106**	**106.00**
	21	34	**34.00**	**33**	34.10
	52	**7**	**7.00**	**7**	**7.00**
	104	**2**	**2.00**	**2**	**2.00**
pr299	222	**84**	**84.00**	**84**	84.60
	445	**29**	**29.90**	30	30.20
	1114	**8**	**8.00**	**8**	**8.00**
	2228	**2**	**2.00**	**2**	**2.00**
d493	170	**54**	**54.00**	**54**	55.30
	340	**18**	**18.00**	**18**	18.40
	851	**5**	**5.00**	**5**	**5.00**
	1700	**2**	**2.00**	**2**	**2.00**
u724	111	**125**	**126.00**	131	132.50
	222	**37**	**37.10**	**37**	38.60
	555	**7**	**7.00**	**7**	7.20
	1110	**2**	**2.00**	**2**	**2.00**
pr1002	650	**111**	**113.40**	118	119.60
	1300	**35**	**36.00**	36	38.10
	3250	**7**	**7.00**	**7**	7.20
	6500	**2**	**2.00**	**2**	**2.00**
pcb1173	120	**149**	**151.30**	158	161.30
	240	**41**	**42.30**	43	44.00
	600	**7**	**7.00**	**7**	7.80
	1200	**2**	**2.00**	**2**	**2.00**
d1291	176	**71**	**73.20**	75	78.80
	352	**23**	**24.00**	25	25.30
	879	**6**	**6.00**	**6**	**6.00**
	1760	**2**	**2.00**	**2**	**2.00**
fl1577	91	**56**	**57.30**	58	60.00
	182	**27**	**27.10**	28	28.40
	456	**8**	**8.00**	**8**	8.80
	910	**2**	**2.00**	**2**	**2.00**

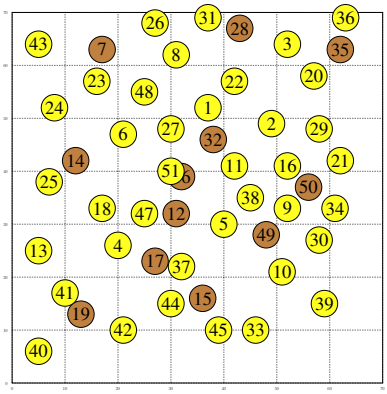
Table 3.4: Summary table

	Least			Average		
Dataset name	$<$	$=$	$>$	$<$	$=$	$>$
OR-Library dataset	9	31	0	14	26	0
TSPLIB dataset	10	29	1	19	21	0
Overall	19	60	1	33	47	0

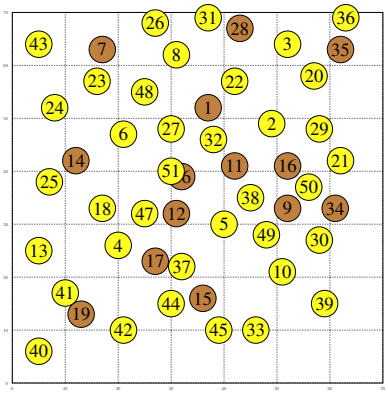
and the same average values as the GA on 24 instances. On the TSPLIB dataset, for the least objective value out of 10 independent runs, out of the 40 instances DDE produced the smaller objective values on 10 instances and the same objective value as GA on 29 instances and only

on one instance DDE has got a higher objective value than GA. On the same TSPLIB dataset, coming to the average objective values of 10 independent runs, DDE produced the smaller average values as compared to GA on 19 instances and the same average values as the GA on 21 instances.

To understand the difference between DACLP solution and ACLP solution, and how this difference varies with R, Fig. 3.2, provides the plots of DACLP and ACLP solutions for R=6, R=15 and R=30 respectively on the *eil51* instance having 51 nodes. In these plots, the nodes selected for locating facilities are depicted in yellow color and nodes not chosen for locating facilities are depicted in brown color. We have not provided the plot for the case with R=3, as both ACLP and DACLP solutions have the same number of facilities which is 50, and only node 46 is a non-facility node. DACLP solutions are obtained through approaches presented here. On the other hand, ACLP solutions were obtained by the approach from the previous chapter. As *eil51* instance is a small instance, all the approaches for ACLP/DACLP obtain the same ACLP/DACLP solutions. As the minimum separating distance R increases, the difference in number of facilities being located is evident as DACLP gives the lower bound on the number of facilities and ACLP the upper bound. It can be observed that for the R=30 case on *eil51* instance, DACLP solution locates 3 facilities in comparison to 6 facilities located by using ACLP, which is 50% lesser number of facilities being located. It can also be observed that in case of ACLP solution, facility nodes tend to be located near the boundaries, whereas in case of DACLP solution, facility nodes tend to be more centrally located.

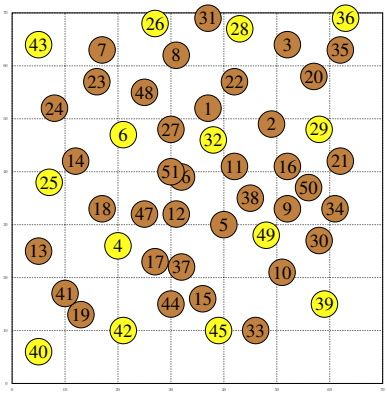


ACLP solution with 39 facilities

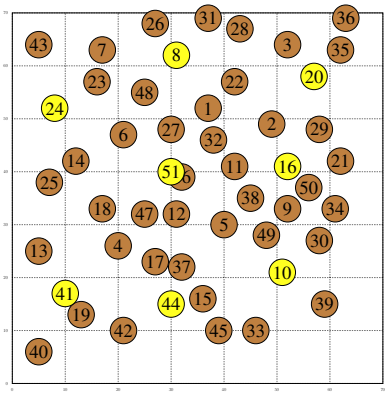


DACLP solution with 37 facilities

(a) R=6

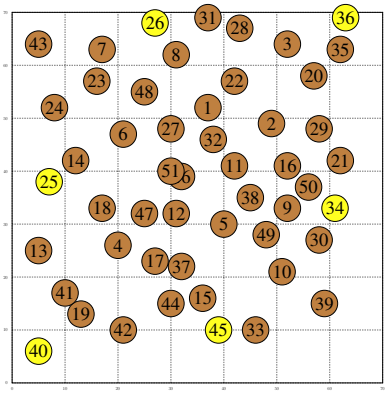


ACLP solution with 14 facilities

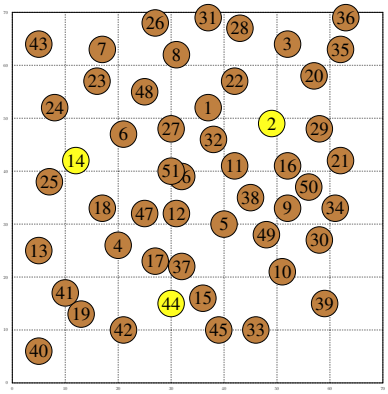


DACLP solution with 8 facilities

(b) R=15



ACLP solution with 6 facilities



DACLP solution with 3 facilities

(c) R=30

Figure 3.2: Plots of ACLP and DACLP solutions on the *eil51* instance having 51 nodes for different values of R

3.3 Weighted ACLP

Given a set of potential facility location sites along with a positive weight associated each and every site as per its importance, weighted ACLP consists in locating a maximum weighted set of facilities in such a manner that no two facilities are within a pre-specified distance of one another. This section is concerned with the weighted version of ACLP while the previous chapter is about the unweighted ACLP.

The weighted ACLP finds its importance in solving many real-world applications. Some of them include but not limited to locating dump yards, nuclear power plants [96, 97], telecommunication equipments, franchise outlets [98, 99], military defence unit location [86], DNA sequence matching [100], forest management [101]. When potential facility location sites have the same importance then unweighted version of ACLP is used, otherwise weighted version of ACLP is used. In most cases, importance of sites differ and weighted version is more appropriate.

Moon and Chaudhry [79] introduced ACLP for the first time in 1984 considering the weighted case and presented an integer programming formulation for this problem. Being the generalization of (unweighted) ACLP considered in previous chapter, the weighted ACLP also belongs to the class of $\mathcal{NP}$-hard problems [79, 80]. Chaudhry et al. [86] proposed four greedy heuristics for weighted version of ACLP, and empirically analyzed the behaviour of these heuristics on instances with upto 50 nodes. It is observed that despite the bad worst-case behavior, these heuristics performed quite well on randomly generated instances.

This section has following subsections. Section 3.3.1 defines the problem formally. Section 3.3.2 and Section 3.3.3 respectively present the proposed discrete differential evolution and genetic algorithm based approaches for the weighted ACLP. Experimental results along with their analysis are presented in Section 3.3.5.

3.3.1 Formal problem definition

Weighted ACLP can be mathematically formulated as an extension of the ACLP presented in previous chapter (Section 2.2) by considering the node weights. Consider a set $V = \{1, 2, \ldots, n\}$ of n potential facility location sites, i.e., $|V| = n$, and a distance R, so that no two facilities can be within distance R of one another. Each site $v \in V$ has an associated positive weight w_v according to the importance of the site and d_{uv} is the shortest distance from site $u \in V$ to site $v \in V$. The forbidden set of site v denoted by Q_v is the set of sites within distance R of site v, i.e., $Q_v = \{u | u \in V \wedge d_{vu} \leq R \wedge v \neq u\}$. The objective of weighted ACLP is to

find a set $V' \subseteq V$ that maximizes $\sum_{v \in V'} w_v$ such that $Q_v \cap V' = \emptyset \; \forall v \in V'$. By introducing binary variables $s_v \forall v \in V$ to indicate whether site v is chosen for locating a facility ($s_v = 1$) or not ($s_v = 0$) and taking a large positive integer M, a mathematical model of weighted ACLP which is originally formulated by Moon and Chaudhry [79] is given below:

$$max\ Z = \sum_{v \in V} w_v s_v \tag{3.5}$$

subject to the following constraints ,

$$M s_v + \sum_{u \in Q_v} s_u \leq M,\ \forall v \in V \tag{3.6}$$

$$s_v \in \{0, 1\},\ \forall v \in V \tag{3.7}$$

Here, equation 3.5 represents the objective function of the weighted ACLP which maximizes the sum of weights of the selected sites. Equation 3.6 specifies that if a facility is located at node v (i.e. $s_v = 1$), then the Ms_v = M, and, as a result $\sum_{u \in Q_v} s_u = 0$. So, it enforces the constraint that if a site v is part of the solution, then all the sites u within the distance R of site v, i.e., all sites belonging to Q_v can not be part of the solution. This constraint is called the neighbourhood adjacency constraint. Clearly, the value of M should be so chosen so that it is larger than $\max_{v \in V}(\sum_{u \in Q_v} s_u)$. Constraint 3.7 enforces the binary nature of decision variables $s_v \forall v \in V$. Few alternative mathematical formulations of weighted ACLP can be found in [81].

3.3.2 DDE approach for WACLP

We have extended our discrete differential evolution (DDE) based approach for ACLP from the Chapter 2 to the weighted version of ACLP also by making required changes while incorporating the weight associated with each node. Subsequent subsections describe the salient features of our DDE approach for the weighted ACLP.

3.3.2.1 Solution encoding and fitness

We have used the bit vector representation to represent a solution just like in the DDE approach for ACLP from the previous chapter (Section 2.3.1). We have used the objective function (equation 3.5) itself as the fitness function.

3.3.2.2 Generating initial population

To generate each member of the initial population of POP_{cnt} candidate solutions, we have extended the randomized greedy method proposed for the ACLP from the Chapter 2 to the weighted ACLP. The only difference is that, in WACLP nodes with highest value of the ratio $\frac{w_v}{|Q_v|}$ are given preference, where w_v is the weight associated with v and Q_v is the forbidden set of node v.

3.3.2.3 DDE framework

The same DDE framework 2.3.3 used for the ACLP has been used for the weighted ACLP also. The same mutation operation from the previous chapter is used in the weighted ACLP case as well (Section 2.3.4). The repair operation which is explained in the subsequent subsection (Section 3.3.2.4) is applied on the mutant to make it feasible and to improve its fitness. In the repair operation, after making the mutant feasible if the latest fitness is within 20% of the best solution's fitness only then we try to improve its fitness. We chose this 20% after large number of experiments to maintain a balance between solution quality and execution time. Then a simple uniform crossover operation as in the ACLP case (Section 2.3.5) is performed between the mutant solution and target solution. Trial solution is repaired and then as per the selection policy (Section 2.3.7) is considered to replace the target solution. In addition, we have implemented a local search (Section 3.3.2.5) to further improve the fitness of the best solution. This process is repeated till the termination criteria is satisfied. And the best solution found over all the iterations is returned as the final best solution.

3.3.2.4 Repair

Repair operation is performed to convert an infeasible solution obtained through mutation / crossover into a feasible solution and then if possible improve its fitness. In the repair operation, first it is checked whether the trial solution is feasible or not by verifying the separating distance constraint. If the trial solution is not feasible, a site v, which has the least value of the ratio $\frac{w_v}{|Q_v|}$ is removed by setting the corresponding bit in the solution to 0. This step is repeated till the trial solution becomes feasible.

Once the trial solution is made feasible, we make an attempt to increase its fitness. To improve the fitness, we will compute the set S_{rem} of all those remaining sites which can still be added to the solution without violating the separating distance constraint. Then, we compute

$Q_u^{'} = Q_u \cap S_{rem}\ \forall u \in S_{rem}$. All those sites v in S_{rem} such that $Q_v^{'} = \emptyset$ are added immediately to the solution by setting the corresponding bits in the solution to 1. If there is no site $v \in S_{rem}$ with $Q_v^{'} = \emptyset$, then a site $v \in S_{rem}$ which has the highest value of $\frac{w_v}{|Q_v^{'}|}$ is made part of the solution by marking the corresponding bit in the solution as 1. The set S_{rem} and the sets $Q_u^{'}$ $\forall u \in S_{rem}$ are updated to reflect the change in configuration of the solution. This process is repeated till S_{rem} becomes empty.

3.3.2.5 Local search

Whenever the best solution changes, a one-one exchange local search is performed on the best solution in an attempt to further improve its fitness. The one-one exchange local search is applied in an iterative manner. We check for each site v in the best solution whether it can be replaced with a site u from its forbidden set Q_v, $u \in Q_v$ having more weight than v, $w_u > w_v$, while satisfying the separating distance constraint. When we find a site v in the best solution which can be replaced with another site u, $u \in Q_v$, we mark the position corresponding to new site u with 1 as being part of the solution and the position corresponding to site v with 0 as being removed from the solution. This procedure is performed once for each site in the best solution, and the fitness of the best solution is updated at every such exchange.

Algorithm 6 provides the pseudo-code for our discrete differential evolution approach, where $u01$ is an uniform variate in [0, 1]. *Mutation*, *Crossover*, *Repair*, f and *Localsearch* are five functions that perform mutation (Section 2.3.4), crossover (Section 2.3.5), repair (Section 3.3.2.4), fitness computation (Section 3.3.2.1) and local search (Section 3.3.2.5) operations respectively.

3.3.3 GA approach for WACLP

We have also developed a steady-state genetic algorithm [66] based approach for weighted ACLP, which is on the same lines as the GA for DACLP presented in Section 3.2.3. We have represented each solution as an ordered list of sites chosen for facility location and the objective function is directly used as the fitness function like in Section 3.2.3.1. Just like in Section 3.2.3.2, the initial solution generation method of our GA for WACLP is also a mix of randomized greedy method used for DDE and purely random method. Only difference is that, in the WACLP nodes having the highest value of the ratio $\frac{w_v}{|Q_v|}$ are given preference. We have utilized probabilistic binary tournament selection method to choose the two parents for crossover

Algorithm 6: DDE algorithm for weighted ACLP

```
Generate initial population;
best_solution ← best solution in initial population;
while (termination condition remains unsatisfied) do
    foreach (target_solution ∈ population) do
        if (u01 ≤ p_m) then
            mutant ← Mutation(best_solution);
            no_mutation ← 0;
        else
            mutant ← best_solution;
            no_mutation ← 1;
        if ((u01 ≤ p_c) or (no_mutation = 1)) then
            trial_solution ← Crossover(mutant, target_solution);
        else
            trial_solution ← mutant;
        trial_solution ← Repair(trial_solution);
        if f(trial_solution) ≥ f(target_solution) then
            target_solution ← trial_solution;
            if f(trial_solution) > f(best_solution) then
                best_solution ← trial_solution;
                best_solution ←Localsearch(best_solution);
return best_solution;
```

and a single parent for mutation like in the Section 3.2.3.3. The parameter ρ_{pbt} governs the probability of selection of the more fit individual in the binary tournament. We have applied crossover and mutation operations in the same manner as in Sections 3.2.3.4, 3.2.3.5 respectively where the corresponding probabilities in crossover are ρ_{add}, ρ_{rand} and mutation probability is ρ_m. Crossover is utilized with probability ρ_c, or else mutation is utilized. Then we perform a local search operation on the child solution to improve its fitness 3.3.4. The same population replacement model as in the GA for DACLP 3.2.3.6 has been used where the newly generated child solution replaces the least fit member of the population subject to the condition that it is distinct from all the current population members. The child is discarded in case it is found to be same as any current population member or worse than the least fit member of the population.

3.3.4 Local search

If the child solution obtained through crossover/mutation is within A% of the best solution then one-one exchange local search is applied iteratively till it is not possible to improve the solution any further.

The pseudo-code for GA has been provided in Algorithm 7, where $BTS()$, $Cross()$, $Mutate()$ and $Localsearch()$ are four functions implementing probabilistic binary tournament selection method, crossover operator, mutation operator and local search (Section 3.3.4) respectively. Further, ps is the population size and $u01$ is a uniform random variate in [0, 1].

Algorithm 7: GA for weighted ACLP

```
Construct ps initial solutions S_1, S_2, ..., S_ps;
S_best ← Best solution among ps initial solutions;
while (termination condition remains unsatisfied) do
    if (u01 < ρ_c) then
        P_1 ← BTS(S_1, ..., S_ps);
        repeat
            P_2 ← BTS(S_1, ..., S_ps);
        until (P_1 ≠ P_2);
        S_C ← Cross(P_1, P_2);
    else
        P_1 ← BTS(S_1, ..., S_ps);
        S_C ← Mutate(P_1);
    if (S_C is within A% of S_best) then
        S_C ← Localsearch(S_C);
    Include S_C in the population as per replacement policy;
    if (S_C is better than S_best) then
        S_best ← S_C;
return S_best;
```

3.3.5 Experimental results

We have implemented both the DDE and GA based approaches for WACLP in C. The parameters used in DDE and their corresponding values are as follows: population size POP_{cnt} = 50, p_m = 0.9, p_c = 0.9, and p_{mut} = 0.02. Likewise, GA parameters and their respective values are as follows: population size ps = 200, ρ_{rand} = 0.75, ρ_{bts} = 0.8, ρ_c = 0.5, ρ_m = 0.75, ρ_{add} = 0.9, and A = 10%. All these parameter values for both DDE and GA are chosen empirically. For each test instance, our approaches are executed 10 times independently on a Linux based 3.40 GHz Core-i5-7500 system with 8 GB RAM. We have also implemented the four greedy heuristics described in Chaudhry et al. [86] and compared their results with the proposed approaches. We denote these four greedy heuristics by H1, H2, H3 and H4 in this chapter. Both DDE and GA based approaches are executed for the same amount of time. On all the instances with number of nodes upto to 100, the DDE and GA based approaches are run for 1 second, on all the instances with number of nodes greater than 100 and upto 500, these two approaches are executed for 2

seconds, and on all the instances with more than 500 nodes, these two approaches are executed for 5 seconds. The execution times of the four greedy heuristics are negligible, and, hence, not reported.

We have used two datasets for evaluating the performance of our approaches, which are derived from Beasley's OR-library[1] and the standard TSPLIB[2] with the number of nodes upto 1577. Both these datasets are first introduced in the context of ACLP in Chapter 2, later modified to include node weights which are random integers between 1 to 10.

The results obtained on the OR library dataset and the TSPLIB dataset are reported in Table 3.5 and Table 3.6 respectively. In these tables the first column, *Instance*, gives the name of the instance. Second column, *R*, represents the minimum separating distance. Columns 3, 4, 5 and 6 represent the best objective value returned by each of the greedy heuristics H1, H2, H3 and H4 respectively. Columns 7 and 8 provide the best and average objective values obtained with the DDE based approach over 10 independent runs, and the average time taken by the DDE approach to reach the best solution over 10 independent runs is given in the 9th column. Similarly columns 10, 11 give the best and average values obtained with the GA based approach over 10 independent runs, and the 12th column provides the average time taken by the GA approach to reach the best solution over 10 independent runs. For each instance, the best results over all the approaches are shown in bold font for easy identification. From these two tables, it can be clearly seen that for all the instances, our proposed approaches based on DDE and GA have performed as good as or better than all the four previously proposed greedy heuristics. Both DDE and GA based approaches obtained the same best and average objective values for instances with upto 100 nodes. For larger instances, the GA based approach performed better than or same as the DDE based approach in terms of both best and average solution quality except for one TSPLIB instance where average solution quality of DDE approach is better. If we look at average time to reach the best solution of the two approaches, we can clearly see that GA converges as fast as or faster than DDE on most of the instances (39 instances out of 40 ORLIB instances and 36 instances out of 40 TSPLIB instances).

Figure 3.3 plots the solutions found by GA for different values of R on instance eil51 with 51 nodes. In this figure, the facilities are shown in green, while the non-facility nodes are shown in blue. This figure clearly show that as the value of R increases, the number of facilities that can be located decreases.

[1] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/esteininfo.html
[2] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html

Table 3.5: Results of DDE, GA and 4 greedy heuristics on OR library dataset

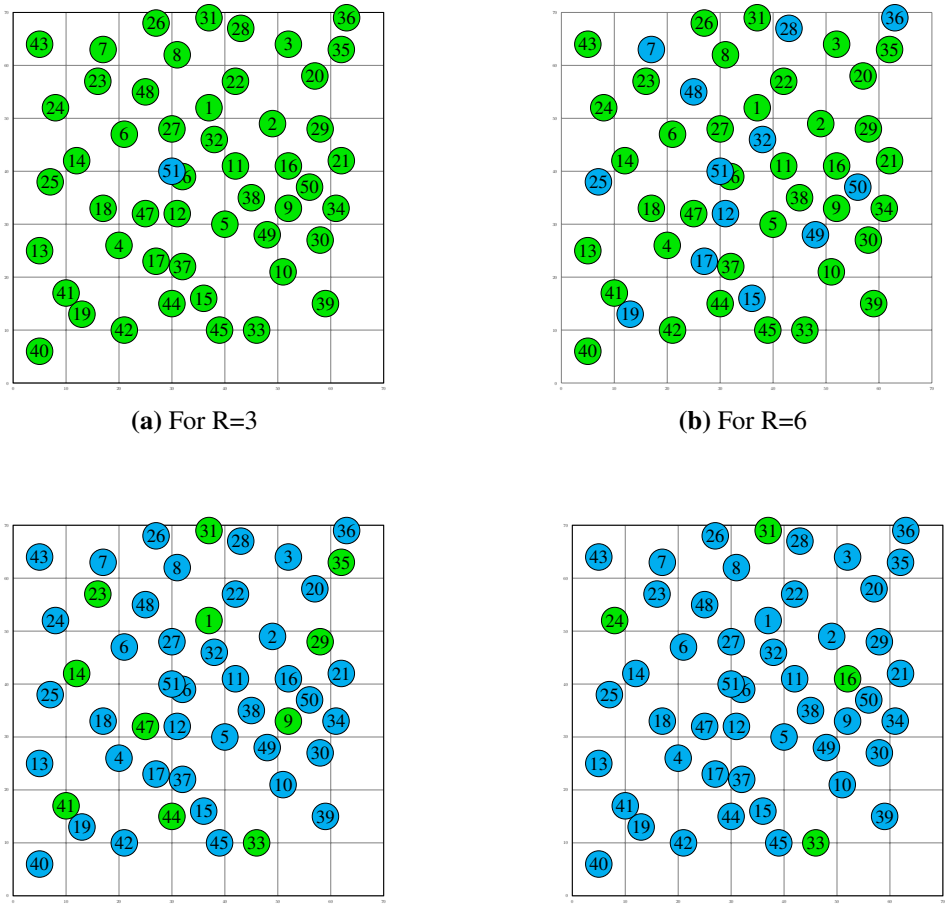
						DDE Based Solution			GA Based Solution		
Instance	R	H1	H2	H3	H4	Best	Average	TTB [a]	Best	Average	TTB [a]
OR_50.1	5	**241**	178	238	**241**	**241**	**241.00**	**0.00**	**241**	**241.00**	**0.00**
	10	**181**	141	177	173	**181**	**181.00**	**0.00**	**181**	**181.00**	**0.00**
	25	**83**	78	69	77	**83**	**83.00**	**0.00**	**83**	**83.00**	**0.00**
	50	37	36	36	37	**38**	**38.00**	**0.00**	**38**	**38.00**	**0.00**
OR_50.2	5	**245**	222	**245**	**245**	**245**	**245.00**	**0.00**	**245**	**245.00**	**0.00**
	10	**209**	188	205	190	**209**	**209.00**	**0.00**	**209**	**209.00**	**0.00**
	25	83	79	82	73	**85**	**85.00**	**0.00**	**85**	**85.00**	**0.00**
	50	**44**	37	36	**44**	**44**	**44.00**	**0.00**	**44**	**44.00**	**0.00**
OR_100.1	5	402	318	399	405	**407**	**407.00**	0.01	**407**	**407.00**	**0.00**
	10	246	240	242	239	**252**	**252.00**	**0.00**	**252**	**252.00**	**0.00**
	25	99	**104**	98	84	**104**	**104.00**	**0.00**	**104**	**104.00**	**0.00**
	50	46	38	**47**	**47**	**47**	**47.00**	**0.00**	**47**	**47.00**	**0.00**
OR_100.2	5	**448**	403	**448**	**448**	**448**	**448.00**	0.01	**448**	**448.00**	**0.00**
	10	**293**	259	**293**	252	**293**	**293.00**	0.01	**293**	**293.00**	**0.00**
	25	105	97	99	94	**107**	**107.00**	**0.00**	**107**	**107.00**	**0.00**
	50	38	**40**	38	38	**40**	**40.00**	**0.00**	**40**	**40.00**	**0.00**
OR_250.1	5	852	751	847	805	**872**	**872.00**	0.15	**872**	**872.00**	**0.02**
	10	403	404	415	405	**434**	**434.00**	0.20	**434**	**434.00**	**0.06**
	25	121	125	126	121	**142**	**142.00**	**0.03**	**142**	**142.00**	**0.03**
	50	46	40	46	46	**52**	**52.00**	0.32	**52**	**52.00**	**0.00**
OR_250.2	5	803	709	801	768	**813**	**813.00**	0.61	**813**	**813.00**	**0.03**
	10	446	417	415	399	**459**	**459.00**	0.02	**459**	**459.00**	**0.01**
	25	127	115	122	124	**132**	131.50	0.22	**132**	**132.00**	**0.02**
	50	48	40	49	48	**56**	**56.00**	0.07	**56**	**56.00**	**0.00**
OR_500.1	5	1215	1079	1179	1090	**1229**	1227.90	1.11	**1229**	**1229.00**	**0.22**
	10	499	477	492	476	**541**	538.60	0.28	**541**	**540.80**	**0.09**
	25	139	126	130	139	**159**	156.30	0.38	**159**	**159.00**	**0.13**
	50	53	40	49	53	**59**	58.70	0.04	**59**	**59.00**	**0.01**
OR_500.2	5	1203	1130	1185	1095	**1246**	1242.60	1.35	**1246**	**1246.00**	**0.46**
	10	519	501	527	516	**564**	**562.70**	0.26	**564**	562.20	**0.07**
	25	148	138	109	147	**156**	153.00	**0.45**	**156**	**156.00**	0.59
	50	57	40	47	57	**59**	**59.00**	0.08	**59**	**59.00**	**0.00**
OR_1000.1	5	1584	1514	1500	1521	1649	1643.60	4.23	**1654**	**1653.80**	**2.79**
	10	615	553	575	610	655	**651.20**	2.13	**657**	**651.20**	**1.01**
	25	158	147	145	154	**171**	168.80	1.11	**171**	**171.00**	**0.19**
	50	**60**	40	56	**60**	**60**	**60.00**	0.02	**60**	**60.00**	**0.00**
OR_1000.2	5	1534	1439	1475	1455	1580	1571.80	4.16	**1587**	**1582.00**	**2.93**
	10	613	576	571	566	**663**	646.20	2.78	**663**	**661.50**	**1.35**
	25	155	146	140	141	**166**	165.30	1.24	**166**	**166.00**	**0.12**
	50	59	40	50	57	**60**	**60.00**	0.05	**60**	**60.00**	**0.01**

[a] Average time to find the best solution in seconds

Table 3.6: Results of DDE, GA and 4 greedy heuristics on TSPLIB dataset

						DDE Based Solution			GA Based Solution		
Instance	R	H1	H2	H3	H4	Best	Average	TTB[a]	Best	Average	TTB[a]
eil51	3	**281**	277	**281**	**281**	**281**	**281.00**	**0.00**	**281**	**281.00**	**0.00**
	6	**233**	200	**233**	**233**	**233**	**233.00**	**0.00**	**233**	**233.00**	**0.00**
	15	**92**	77	90	77	**92**	**92.00**	0.01	**92**	**92.00**	**0.00**
	30	37	**40**	**40**	35	**40**	**40.00**	**0.00**	**40**	**40.00**	**0.00**
rat99	8	**501**	422	498	498	**501**	**501.00**	0.01	**501**	**501.00**	**0.00**
	15	305	281	302	277	**307**	**307.00**	0.06	**307**	**307.00**	**0.00**
	38	104	96	103	94	**105**	**105.00**	**0.00**	**105**	**105.00**	**0.00**
	75	43	45	45	40	**46**	**46.00**	**0.00**	**46**	**46.00**	**0.00**
rat195	11	**803**	733	794	783	**803**	**803.00**	0.10	**803**	**803.00**	**0.01**
	21	395	360	391	372	**417**	**417.00**	0.10	**417**	**417.00**	**0.01**
	52	121	104	100	117	**130**	129.70	0.68	**130**	**130.00**	**0.00**
	104	55	**56**	54	55	**56**	**56.00**	**0.00**	**56**	**56.00**	**0.00**
pr299	223	811	734	790	759	**816**	813.90	0.38	**816**	**816.00**	**0.02**
	446	393	374	367	352	**405**	404.40	0.57	**405**	**405.00**	**0.11**
	1114	117	106	111	98	**131**	130.40	0.02	**131**	**131.00**	**0.00**
	2228	46	39	46	47	**55**	53.20	**0.00**	**55**	**55.00**	**0.00**
d493	170	628	578	608	593	**651**	650.00	0.66	**651**	**650.90**	**0.57**
	340	272	227	248	257	**285**	284.40	0.55	**285**	**285.00**	**0.09**
	851	79	59	71	76	81	81.00	**0.00**	**82**	**82.00**	0.10
	1700	33	29	33	33	**34**	**34.00**	**0.00**	**34**	**34.00**	**0.00**
u724	111	1409	1296	1369	1280	1464	1459.60	3.98	**1465**	**1464.60**	**0.81**
	222	566	567	531	536	603	600.90	**1.89**	**604**	**601.90**	2.18
	555	145	125	130	151	**157**	152.00	1.01	**157**	**157.00**	**0.35**
	1110	58	49	56	48	**59**	**59.00**	0.01	**59**	**59.00**	**0.00**
pr1002	650	1398	1290	1348	1328	1442	1432.70	4.62	**1449**	**1447.70**	**2.98**
	1300	550	497	527	530	**592**	587.60	4.29	**592**	**592.00**	**0.75**
	3250	152	116	132	140	**159**	156.80	0.77	**159**	**159.00**	**0.05**
	6500	54	40	48	54	**58**	57.80	**0.09**	**58**	**58.00**	0.48
pcb1173	120	1707	1633	1682	1668	1795	1773.30	4.53	**1829**	**1823.50**	**4.29**
	240	625	569	595	567	671	666.40	3.81	**677**	**672.70**	**3.51**
	600	159	119	134	140	**169**	166.20	0.59	**169**	**169.00**	**0.06**
	1200	57	40	50	49	**60**	**60.00**	0.01	**60**	**60.00**	**0.00**
d1291	176	1011	999	972	938	1075	1066.90	3.89	**1076**	**1071.30**	**2.56**
	352	388	351	389	338	426	424.00	1.57	**427**	**425.50**	**0.74**
	879	121	100	112	112	**122**	**122.00**	0.02	**122**	**122.00**	**0.01**
	1760	44	40	44	43	**46**	44.60	**0.04**	**46**	**46.00**	1.07
fl1577	91	790	723	734	683	808	804.80	3.28	**818**	**817.20**	**2.30**
	182	404	363	381	367	439	438.60	2.71	**444**	**441.50**	**1.46**
	456	130	110	117	107	**147**	**147.00**	0.85	**147**	**147.00**	**0.05**
	910	58	**60**	58	49	**60**	**60.00**	**0.00**	**60**	**60.00**	**0.00**

[a] Average time to find the best solution in seconds



(a) For R=3 **(b)** For R=6

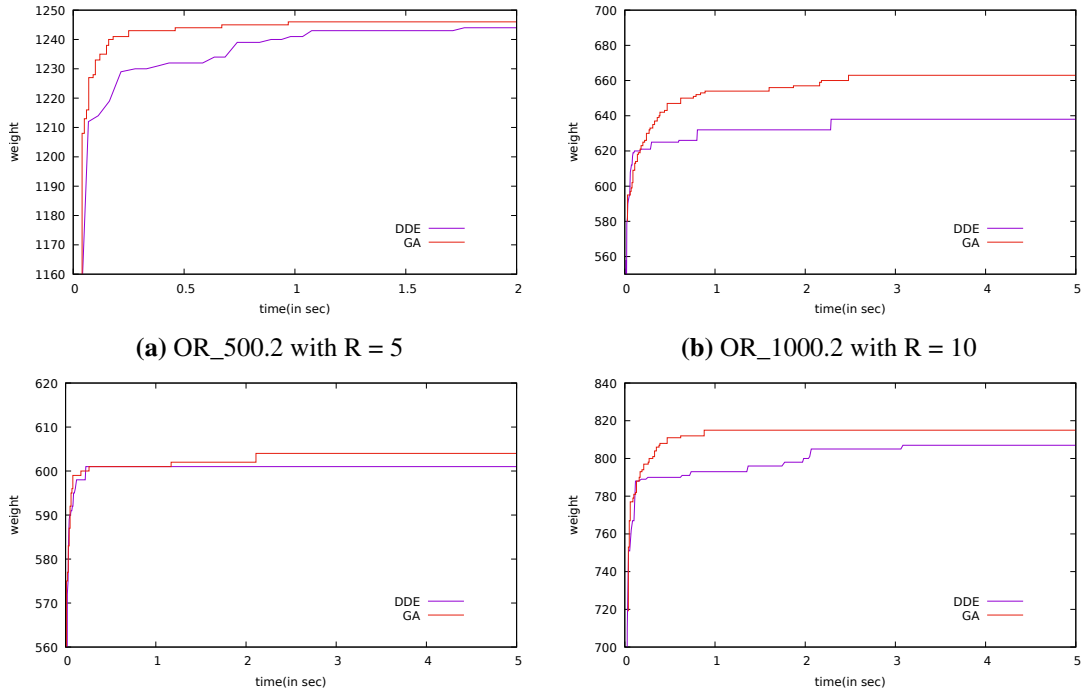
(c) For R=15 **(d)** For R=30

Figure 3.3: Weighted ACLP solutions found by GA for different R values on *eil51* instance

Figure 3.4 shows the convergence behavior of DDE and GA on 4 different instances. On all the 4 instances, GA converges faster than DDE. In fact, this is the case on all the instances.

We have conducted the Wilcoxon signed-ranks test [84] to check the statistical significance of the difference in results obtained by the proposed approaches based on DDE and GA. We performed the two tailed Wilcoxon signed ranks test with significance level set to 0.01 (i.e. p-value ≤ 0.01) using the calculator available online[1]. The test is performed separately on OR library dataset and the TSPLIB dataset. In the Table 3.7, the column N represents the number of instances considered, NWT is the number of instances without tie, W^+ is the sum of ranks of the instances where DDE performed better whereas W^- is the sum of ranks of the instances where GA performed better. The test statistic T is the minimum of W^+ and W^-. For the OR

[1] `https://mathcracker.com/wilcoxon-signed-ranks.php`



(a) OR_500.2 with R = 5

(b) OR_1000.2 with R = 10

(c) u724 with R = 222

(d) fl1577 with R = 91

Figure 3.4: Covergence behavior of DDE and GA on 4 different instances

library dataset the value of $T = 2$ is less than the corresponding T^* value of 12 and for the TSPLIB dataset $T = 0$ is less than the corresponding T^* value of 62. This shows the better performance of GA based approach over the DDE approach is statistically significant.

Table 3.7: Wilcoxon signed-ranks test

	N	NWT	W^+	W^-	T	T^*	Significant
OR library	40	13	2	89	2	12	yes
TSPLIB	40	23	0	276	0	62	yes

3.4 Conclusions

Both DACLP and WACLP are understudied problems. In this chapter, we have proposed two population based metaheuristics for DACLP and WACLP, viz. a discrete differential evolution (DDE) based approach and a genetic algorithm (GA) based approach. Both DDE and GA fall under the broad class of evolutionary algorithms. We have tested the performance of our approaches on a total of 80 instances with upto 1577 sites.

For DACLP, when the least objective value is considered, DDE produced smaller objective values than GA on 19 instances, same objective values on 60 instances, and a greater objective value on 1 instance. Similarly, for the average objective value, DDE produced smaller objective values on 33 instances, equal objective values on 47 instances. Overall, when the comparison is done with respect to least objective value DDE produced solutions of equal or better quality in comparison to GA on 79 out of the 80 instances and when the comparison is done with respect to average objective value on all the 80 instances DDE produced solutions of equal or better quality in comparison to GA.

For WACLP, computational results show that our metaheuristic approaches performed as good as or better than the four greedy heuristics available in the literature on all the instances, and GA based approach performed as good as or better than the DDE based approach on 79 out of the 80 instances.

In the case of DACLP, a dedicated repair operation applied in DDE contributed to its better performance by removing additional facilities and that too in a diverse manner while making the solution proper thereby paving the way for a better exploration of the search space. Hence, DDE performed better than GA for DACLP. In the case of WACLP, it is a maximization problem with positive weights associated with each node. In our proposed GA for WACLP, the exhaustive 1-1 exchange local search operation performed on the best solution contributed to its better performance as compared to the DDE.

Chapter 4

Obnoxious cooperative maximum covering location problem

4.1 Introduction

Solving a location problem involves locating one or more facilities in the given solution space while optimizing the pre-specified criteria. In location science, considering the facilities and demand points the most common criteria is the interaction between a facility and the demand points which interact with that facility. The facilities can be desirable, semi-obnoxious or obnoxious. Locating facilities such as schools, hospitals, banks, supermarkets are the examples of desirable facilities where it is beneficial to have these facilities close to the demand points or customers [4, 102]. On the other hand, facilities such as nuclear power plants, prisons, dump yards, military installations and industrial facilities causing pollution are examples of obnoxious facilities, which even though are required for the society, but produce a negative or undesirable effect[103, 104]. So, the problem of locating the obnoxious facilities also needs to be carefully addressed.

The cooperative maximum covering location problem (CMCLP) [2, 18, 43] is an example of cooperative coverage model where all facilities contribute to the coverage of each demand point and it is concerned with locating a given p number of desirable facilities so as to maximize the total demand covered. But when the facility location problem is concerned with locating undesirable or obnoxious facilities under the cooperative coverage model, then the objective is to maximize the total uncovered demand. This variant of the problem is called obnoxious cooperative maximum covering location problem (OCMCLP) [2]. CMCLP and OCMCLP are

highly complex facility location problems because of the use of the cooperative coverage model and allowing the facilities to be located along the edges joining the demand points in addition to demand points themselves. For the CMCLP, a hybrid artificial bee colony approach was proposed in [43]. However, no metaheuristic approach exists for the OCMCLP.

In all of the applications of locating obnoxious facilities, the nuisance generated by an obnoxious facility decreases over distance following some signal strength function. When more than one of such obnoxious facilities are located, the nuisance effect on the demand points is cumulative. For example, the poor air quality at a community will be a combined effect of several industrial facilities in the locality which release polluting chemicals into the the air rather than just due to the nearest industrial facility. Even though the nearest industrial facility has the higher contribution of the negative effect, the effect of other such industries which are little farther than the closest facility can't be ignored. But many of the existing models consider solving for locating a single obnoxious facility, hence no cumulative effect is taken into account [105, 106, 107, 108].

Church and Garfinkel [103] first introduced the obnoxious facility location problem in 1978. It is also a well studied problem in the location literature. We refer the interested readers to [104, 109, 110, 111, 112]. Melachrinoudis [26] and Drezner, Kalczynski and Salhi [27] present the reviews of recent obnoxious models.

There are several different ways in which obnoxious facility location models can be formulated. Maximizing the minimum distance of the demand points from the obnoxious facilities is the most common formulation as presented in [103, 113]. Another formulation is to consider the negative effect of the facilities declining by the square of the distance between facilities and demand points [105]. There are single facility locating models [114] and multiple facilities locating models [27, 115]. Drezner et al. [116] also proposed the Weber obnoxious facility location model which is formulated from the classic Weber location problem [117, 118, 119] by adding an additional condition that the facility must be placed at least at a given distance from the demand point because the facility is obnoxious.

Even though several researchers worked on the obnoxious facility location problem, the cooperative coverage model of the problem has not received much attention. Averbakh et al. [2] have used the cooperative coverage model for the obnoxious facility location problem and formulated the OCMCLP where it is possible to locate the facilities both at the nodes and along the edges. Two greedy heuristics (G1, G2) and two interchange heuristics (I1, I2) were also proposed in [2] to solve the OCMCLP. In the first greedy heuristic (G1), beginning with an

empty solution, at each step a new facility is added to the solution so that a facility which will cover as few demand points as possible is selected to be part of the solution. In the second greedy heuristic (G2), at a time a pair of facilities are added to be part of the solution till the required number of p facilities are located. In G2, if the value of p is odd then a single facility is added at the end. In the first interchange heuristic I1, the solution generated by G1 is considered as the initial solution S1 and a local search is performed so as to relocate one facility at a time from S1 till there is no further improvement. If no improvement is possible with one facility relocation, then they try to relocate two facilities at a time from S1 till no further improvement is possible. The second interchange heuristic I2 works in the same manner as I1 except that the solution generated by G2 is considered as the initial solution. These are the only approaches available in the literature for OCMCLP. After Averbakh et al. [2], Drezner et al. [120] worked on an obnoxious facility location problem under cooperative coverage model where locations of facilities were restricted to a finite area, no concept of threshold was used to consider a demand point as covered/uncovered, and the objective was to minimize the maximum cumulative nuisance at any demand point. A Voronoi-based heuristic was proposed to solve this problem.

In this chapter, we have proposed a genetic algorithm (GA) based approach for solving the OCMCLP as formulated in [2]. Over the last several decades, genetic algorithm (GA) has been used for solving innumerable combinatorial optimization problems in various domains. Some recent GA based approaches for addressing combinatorial optimization problems can be found in [90, 91, 121, 122, 123, 124]. Since there exists no metaheuristic based approaches in the literature to solve the OCMCLP, this served as the motivation to develop a GA based approach to solve the OCMCLP. Our approach makes use of appropriate problem specific information in genetic operators as well as in local search. We have evaluated the performance of the proposed approach on the same test instances as used in [2] and compared the results obtained with two interchange heuristics presented in [2]. These comparisons clearly demonstrate our proposed approach to be superior.

The rest of this chapter is organized as follows: Section 4.2 provides a formal definition of the OCMCLP. Section 4.3 presents the proposed GA approach for the OCMCLP. Computational results and their analysis are presented in Section 4.4. The last section, viz. Section 4.5 presents some concluding remarks.

4.2 Formal problem definition

To formally define the OCMCLP, we have used the same notational conventions as used in [2]. Important notational conventions are also summarized in Table 4.1. Let $G = (V, E)$ be an undirected graph with $V = (1, ..., n)$ being the set of demand points and $E = (e_1, ..., e_m)$ being the set of edges connecting various demand points. A non-negative real weight w_i associated with each demand point i is given indicating the total demand at this point. l_k is the length of each edge e_k. We need to locate a total of p facilities either at the demand points or along the edges joining these demand points in order to cover these demand points. No two facilities can be located at the same point. A demand point is deemed covered if the cumulative signal strength from all the p facilities received at that point is not less than a threshold T. The OCMCLP is concerned with the obnoxious facilities and it seeks to find a location vector X_p of p facilities such that the sum total of the weights of the uncovered demand points is maximized.

$$f(X_p, T) = \sum_{i:\ \Phi_i(X_p)<T} w_i \tag{4.1}$$

The overall signal strength $\Phi_i(X_p)$ at a demand point $i \in V$ is the sum of the signal strengths of all the signals received by i from the p facilities, i.e.,

$$\Phi_i(X_p) = \sum_{k=1}^{p} \phi(d_i(x_k)) \tag{4.2}$$

Given an edge e_k joining demand point j_1 with j_2 ($e_k = (j_1, j_2)$), an ordered pair (e_k, r) is used to represent the location x of a facility along e_k, where r is the relative distance of x from j_1 with respect to length l_k of edge e_k, i.e., $r \in [0, 1]$. The distance to this facility from a demand point $i \in V$ can be computed in the following manner

$$d(i, x) = \min\{d(i, j_1) + r \times l_k, d(i, j_2) + (1 - r) \times l_k\} \tag{4.3}$$

Where $d(i, j_1)$ is the length of the shortest path between demand point i and demand point j_1. Likewise, $d(i, j_2)$ is the length of the shortest path between demand point i and demand point j_2.

We will now explain OCMCLP with the help of an example. This example uses the network shown in Figure 4.1. There are a total of 9 demand points and the weight associated with

Table 4.1: Important notational conventions

Notation	Meaning
n	Number of demand points or vertices, i.e., $n = \|V\|$
w_i	Weight of demand point $i \in V$
l_k	Length of each edge $e_k \in E$
X	The location space
p	Number of facilities that need to be located
X_p	Location vector containing locations for p facilities
$d_i(x_j)$	Distance between demand point $i \in V$ and facility j
$\phi(d)$	Signal strength at distance d from the facility, $\phi(d) = \max\{0, 1 - d/U\}$
$\Phi_i(X_p)$	Overall signal strength at demand point $i \in V$
U	Distance at which signal strength becomes zero
T	Minimum threshold value for coverage

each demand point is mentioned next to the demand point number in the Figure 4.1 and edge lengths are mentioned along the edges. For the sake of this example, we have taken the signal strength function $\phi(d) = \max\left\{0, 1 - \frac{d}{10}\right\}$, the number of facilities to be located p = 3, and the minimum threshold value of the overall signal strength for coverage T = 0.3.

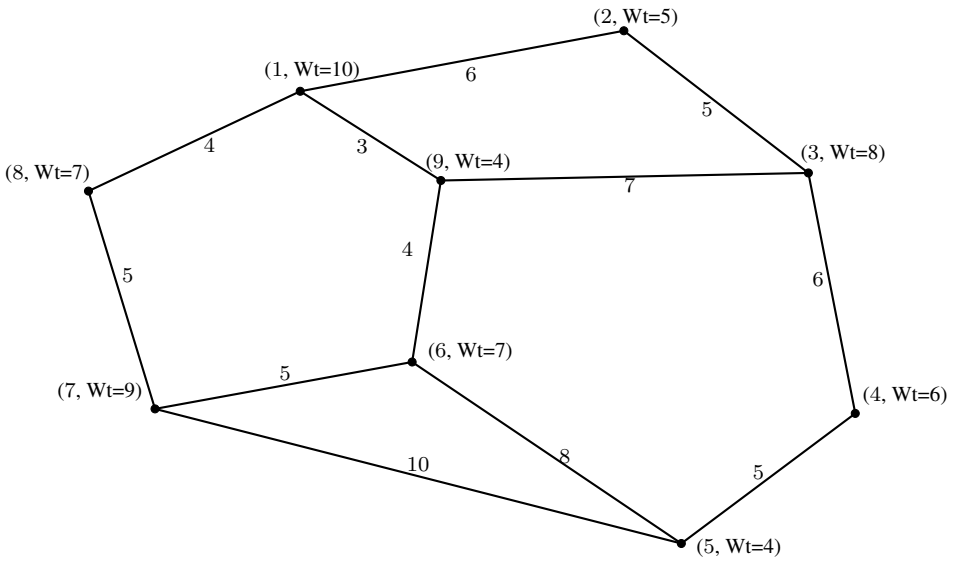


Figure 4.1: A sample network used for explaining OCMCLP

One feasible solution for the OCMCLP is $X = \{(5), ((5,7), 0.2), ((3,9), 0.3)\}$, where first facility is located at demand point 5, second facility is located at a relative distance of $r = 0.2$ from demand point 5 on the edge $(5,7)$, and the third facility is located on edge $(3,9)$ at a

relative distance of $r = 0.3$ from demand point 3. So the demand points $3, 4, 5, 6$, and 9 are covered as the cumulative signal strength received at each of these demand points from the three facilities are greater than the threshold $T = 0.3$. At the remaining demand points, the sum of the signal strength received from the three facilities is less than the threshold, hence they are uncovered. So, this solution produces an objective value of 31, which is the sum of weights of all the uncovered demand points, viz. $1, 2, 7$, and 8. For the sake of illustration, we have picked the points for locating facilities randomly in this example. There may exist some other solutions giving better objective values than this solution for this network.

4.3 Proposed steady-state genetic algorithm approach

This section presents our steady state genetic algorithm approach for OCMCLP. Each feature of our approach has been presented in a separate subsection.

4.3.1 Solution encoding

Each solution is represented as a set of p locations where facilities can be placed. We have used ordered list representation for sets as it helps in implementing the crossover operator and uniqueness checking of a solution in an efficient manner.

To facilitate ordered list representation for potential facility locations, we have ordered the endpoints of an edge by their indices, i.e., if e_k is an edge joining demand points j_1 and j_2 with $j_1 < j_2$ then $e_k = (j_1, j_2)$. This is done so that each location along an edge is represented uniquely. Now, each potential location on an edge $e_k = (j_1, j_2)$ is represented as a triplet (j_1, j_2, r), where r is same as explained in Section 4.2. Each potential location on a demand point j_1 is also represented as a triplet, but in the form (j_1, j_1, 0). This is done to make representation of a demand point unique. Now, a location (x_1, y_1, z_1) precedes another location (x_2, y_2, z_2) only when either ($x_1 < x_2$) or (($x_1 = x_2$) and ($y_1 < y_2$)) or ((($x_1 = x_2$) and ($y_1 = y_2$) and ($z_1 < z_2$)

4.3.2 Fitness evaluation

We have utilized two fitness functions like [2] for evaluating the fitness of a solution. Objective function $f(X_p, T)$ itself (as defined in 4.1) is taken as the primary fitness function. Sum total of signal strengths at all demand points, i.e., $\sum_{i \in V} \Phi_i (X_p)$ is taken as the secondary fitness

function. The secondary fitness function is required to identify a better solution among solutions that have the same value for the primary fitness function. We consider a solution S_1 to be better than another solution S_2, if the solution S_1 either has a larger primary fitness function value, i.e.,

$$f(S_1, T) > f(S_2, T) \tag{4.4}$$

or, if the primary fitness function values of solution S_1 and solution S_2 are equal, and solution S_1 yields smaller sum total of signal strengths at all demand points, i.e.,

$$\sum_{i \in V} \Phi_i (S_1) < \sum_{i \in V} \Phi_i (S_2) \tag{4.5}$$

Actually, several solutions can have the same primary fitness function value and secondary fitness function is used to break the tie. Obviously, among the solutions having the same value of the primary fitness function, a solution having smaller sum total of signal strengths at all demand points is more likely to produce even better solutions when subjected to crossover and mutation.

4.3.3 Generating the initial population of solutions

We have used a randomized greedy approach to generate the initial population POP_SIZE number of solutions. To generate an initial solution a subset of p locations needs to be selected such that the weighted sum of uncovered demand points is maximized. In the proposed method, we start with an empty solution X_p and facilities are added one by one till we get the required p facilities. For this, we find the set of points S which contains all the nodes in V, and also points along each edge. Points along each edge are considered at intervals of 0.005 relative distance from one another. On each edge, there are infinitely many possible locations where facilities can be located. After several experiments, we have empirically chosen the relative distance of 0.005 between possible locations. Let us say $x, y \in V$ and (x, y) is an edge in the graph G with edge length l. We add x, y to the set S and all the points at $x + 0.005 * l$, $x + 0.01 * l$, $x + 0.015 * l$ and so on till $x + 0.995 * l$ are also added to the set S. For each point in the set S, we calculate the amount of coverage it provides to the yet uncovered demand points. The objective value corresponding to each point $i \in S$ is calculated as the sum of the weights of all the uncovered points whose signal threshold requirement T is greater than the signal received from i. After evaluating all the points in the set S, best Y points are chosen. One point is randomly selected

from the set of Y best points to be added to the partial solution X_p. Once a location from S is added to the solution X_p, it is deleted from S and the thresholds for all the demand points are updated using the following equation 4.6.

$$T_i\left(X_p\right) = max\left\{0, T - \Phi_i\left(X_p\right)\right\} \forall i \in V \tag{4.6}$$

This iterative procedure is repeated till we get the p facilities in the solution X_p. Once a solution is generated, we check whether it is a unique solution compared to the already generated population of initial solutions. The newly generated solution is added to the population of solutions only if it is unique, otherwise it is discarded. This iterative process continues till POP_SIZE solutions are generated.

4.3.4 SSGA framework

Given the initial population of solutions, the steady state genetic algorithm based approach for solving the OCMCLP follows an iterative process. In each iteration, we apply crossover and mutation in a mutually exclusive manner. Crossover is applied with probability P_c and mutation is applied with the remaining probability of $1 - P_c$. On the child solution generated after crossover or mutation, we have applied a local search to improve its fitness. Following the local search, population replacement method is applied. Then the next iteration begins. This is repeated for all the MAX_ITERS iterations. At the end of the MAX_ITERS iterations, the best solution found since the beginning of the algorithm is returned as the final solution found by the algorithm.

4.3.5 Selection

We need to select two parents from the population to perform crossover and a single parent to perform mutation operation. We have used the probabilistic binary tournament selection method for this. The parameter P_{bts} governs the probability of selection of the more fit individual in the binary tournament. Two solutions are selected uniformly at random from the population and their fitness is compared. The solution with higher fitness between the two is chosen with the probability P_{bts}, or else the lower fitness solution between the two is chosen, i.e., the worse solution between the two is chosen with probability 1-P_{bts}. In general, probabilistic binary tournament selection performs better than roulette wheel selection and has the same performance

as rank selection, while being computationally much less expensive than latter two methods [125]. This is the reason for using probabilistic binary tournament selection in our approach.

4.3.6 Crossover

In the crossover, we select two solutions which act as parents using binary tournament selection method. To generate a child solution, common facilities from the two selected parents are copied to the child. As we have used ordered list representation, common facilities can be found in $\mathcal{O}(p)$ time. Let us say there are c common facilities, they are copied to the child. Since each solution must have p facilities, therefore, to add the remaining $p - c$ facilities, we follow the same randomized greedy approach as used in the initial population generation. Using this method, the remaining facilities are added one by one till we get total p facilities in the resulting solution.

4.3.7 Mutation

From the population, a solution is selected using binary tournament selection. From this selected solution, facilities are removed with probability P_m. For each facility under consideration a uniform random number r is generated between 0 and 1. If r is greater than P_m that facility is copied to the resulting mutant solution. On the other hand, if r is less than or equal to P_m that facility is not copied to the mutant. Let us say k facilities are copied to the mutant in this manner. Since each solution must have p facilities, to add the remaining $p - k$ facilities, we follow the same randomized greedy approach as used in the initial population generation. Using this method, the remaining facilities are added one by one till we get total p facilities in the mutant solution.

Our crossover operator retains the common facilities between the two selected parents to build a better child solution by making use of these facilities. If mutation is applied after crossover then some of these facilities may get deleted from the child solution. Hence, crossover and mutation have been used in a mutually exclusive manner. During each iteration, crossover is applied with probability P_c, or else mutation is applied.

4.3.8 Local search

On the solution X_C generated through crossover or mutation, a one-one exchange local search operation is applied. This local search is derived from the local search proposed in [43]. As part

of the local search, we have considered each facility $t \in X_C$ one-by-one in an iterative manner and a point b_t to relocate it is determined. For this also, we followed the randomized greedy approach as in the initial solution generation where we compute the set of Y best points for solution $(X_C \setminus \{t\})$ and a point b_t is randomly selected from these Y points to be part of the solution. If the solution $(X_C \setminus \{t\}) \cup \{b_t\}$ has higher fitness in comparison to X_C then X_C is replaced with this new solution. This process is redone for all the facilities $t \in X_C$.

4.3.9 Population replacement model

Our GA uses steady state population replacement model. In this model, only a single child solution is constructed in each generation. If the child solution is unique from the currently existing members of the population then its fitness is compared with the worst solution in the population. And the child solution replaces the worst solution of the population if it has better fitness. On the other hand if the child solution is found to be the same as any current population member or even though it is unique but its fitness is less than the least fit member of the population, in both these cases the child solution is discarded.

Algorithm 8 provides the pseudo-code for our genetic algorithm approach where $u01$ is a uniform variate in [0, 1]. *BTS*, *Crossover*, *Mutation*, and *Local_search* are four functions that perform binary tournament selection (Section 4.3.5), crossover (Section 4.3.6), mutation (Section 4.3.7), and local search (Section 4.3.8) operations respectively.

Algorithm 8: GA for OCMCLP

```
Construct POP_SIZE initial solutions X_1, X_2, ..., X_POP_SIZE;
X_best ← Best solution among POP_SIZE initial solutions;
while (termination condition remains unsatisfied) do
    if (u01 < P_c) then
        P_1 ← BTS(X_1, ..., X_POP_SIZE);
        P_2 ← BTS(X_1, ..., X_POP_SIZE);
        if (P_1 == P_2) then
            P_2 ← Get the next solution after P_2 (modulo population size) in the population.
        X_C ← Crossover(P_1, P_2);
    else
        P_1 ← BTS(X_1, ..., X_POP_SIZE);
        X_C ← Mutation(P_1);
    X_C ← Local_search(X_C);
    if (X_C is better than X_best) then
        X_best ← X_C;
    Include X_C in the population as per replacement policy;
return X_best;
```

4.4 Computational results

We have compared the performance of our GA based approach with the state-of-the-art approaches available for OCMCLP which are discussed in [2]. For the fairness of comparison, we have used the same datasets and the same values for various parameters as in [2]. Averbakh et al. [2] randomly generated five instances of datasets for a given combination of the number of nodes n and average degree of nodes dgr. The values of n and dgr belong to the following set of values $n \in \{40, 60, 80, 100, 120, 140, 160, 180, 200\}$ and $dgr \in \{5, 6, 7\}$ respectively. On each dataset, we have executed our approach with three different values for the number of facilities to be located, $p = 3, 4, 5$. And the three different values of threshold value for the signal strength, T, are considered as $T = 0.1, 0.3, 0.5$ as in [2]. We have taken the linear signal strength function of $\phi(d) = \max\{0, 1 - d/U\}$ and the parameter U was determined as a fraction of the network diameter where $U_{\%} = 0.65, 0.75, 0.85$ for $T = 0.1, 0.3$ and $U_{\%} = 0.7, 0.8, 0.9$ for $T = 0.5$ just as mentioned in [2] .

We have implemented our GA based approach in C and executed it on a Linux based Intel Core i5 8600 system with 8 GB memory running at 3.10 GHz. The parameters specific to GA and their corresponding values are as follows: $POP\_SIZE = 100$, $MAX\_ITERS = 200$, $Y = 10$, $P_c = 0.5$, $P_m = 0.1$, $P_{bts} = 0.7$. These parameter values are chosen based on empirical observations spanning over a number of trials.

In [2], Averbakh et al. proposed two interchange heuristics, I1 and I2, for solving the OCMCLP. The results of our GA based approach are compared with these two interchange heuristics I1 and I2. We have reported the results of our GA approach along with I1 and I2 approaches in 9 tables, viz. Table 4.2, Table 4.3, Table 4.4, Table 4.5, Table 4.6, Table 4.7, Table 4.8, Table 4.9, and Table 4.10. Each of these tables present the results for a particular combination of the parameters T and $U_{\%}$. Within each table, for a given dataset of size n we have shown the average of objective values obtained for all the three different degrees 5, 6 and 7. We have obtained the data for I1 and I2 via e-mail from the corresponding author of [2]. In all these nine tables (Table 4.2–Table 4.10), the first column gives the number of facilities being located, p. The second column gives the number of nodes in the graph, n. Third and fourth columns give the objective value and time taken by I1 while fifth and sixth columns give the objective value and time taken by I2 and the seventh and eighth columns give the objective value and time taken by GA respectively. All times are in seconds.

Table 4.2: Comparison of I1 and I2 interchange heuristics of [2] with GA on instances with $T = 0.1$, $U_{\%} = 0.65$

		I1		I2		GA	
	n	Obj.	Time	Obj.	Time	Obj.	Time
	40	332.67	**0.10**	336.60	0.20	**337.60**	3.06
	60	494.93	**0.10**	**496.20**	0.20	495.40	6.61
	80	658.87	**0.20**	**660.27**	0.40	**660.27**	11.35
	100	783.27	**0.30**	783.53	1.10	**799.73**	15.55
$p = 3$	120	913.87	**0.70**	920.47	2.00	**946.00**	21.22
	140	1029.00	**2.00**	1042.40	3.20	**1071.13**	28.02
	160	1141.40	**2.90**	1146.07	4.60	**1185.00**	35.58
	180	1357.53	**3.70**	1353.07	5.90	**1392.40**	45.63
	200	1493.73	**5.90**	1495.53	9.30	**1533.33**	55.96
	40	329.20	**0.10**	**332.47**	0.30	332.13	3.88
	60	488.47	**0.10**	489.27	0.20	**491.73**	8.29
	80	642.27	**0.30**	645.13	0.60	**647.27**	14.18
	100	777.07	**0.60**	777.07	1.30	**781.00**	19.19
$p = 4$	120	907.67	**1.20**	912.53	2.50	**922.60**	26.56
	140	1023.27	**3.70**	1037.73	4.80	**1056.00**	35.52
	160	1122.47	**3.10**	1129.27	6.10	**1154.33**	44.05
	180	1352.00	**5.70**	1351.00	9.00	**1367.00**	56.41
	200	1478.53	**7.90**	1490.53	13.40	**1507.40**	67.78
	40	326.87	**0.20**	329.40	0.30	**332.13**	4.75
	60	484.60	**0.10**	485.20	0.30	**489.33**	10.02
	80	640.47	**0.40**	641.60	0.70	**642.73**	17.17
	100	773.40	**0.90**	771.67	1.90	**775.60**	23.10
$p = 5$	120	897.80	**1.70**	908.00	3.90	**915.73**	32.03
	140	1017.40	**4.20**	1030.87	6.00	**1046.67**	42.11
	160	1122.47	**4.50**	1121.93	6.60	**1140.53**	52.51
	180	1342.00	**7.30**	1337.33	11.60	**1359.00**	66.96
	200	1474.27	**10.90**	1485.33	15.90	**1492.33**	81.29

Table 4.3: Comparison of I1 and I2 interchange heuristics of [2] with GA on instances with $T = 0.1$, $U_{\%} = 0.75$

		I1		I2		GA	
	n	Obj.	Time	Obj.	Time	Obj.	Time
	40	262.07	**0.00**	**266.60**	0.10	265.73	2.87
	60	358.93	**0.00**	**360.93**	0.10	359.53	5.68
	80	461.73	**0.10**	**467.60**	0.20	457.53	9.70
	100	545.20	**0.30**	545.20	0.50	**557.47**	13.11
$p = 3$	120	593.07	**0.40**	596.93	0.90	**618.00**	16.85
	140	611.40	**1.30**	615.87	1.60	**664.93**	22.06
	160	639.67	**1.50**	640.93	1.80	**683.47**	26.63
	180	746.07	**2.00**	752.00	3.00	**817.33**	33.81
	200	845.07	**2.60**	849.53	3.90	**909.20**	41.44
	40	257.67	**0.00**	258.87	0.10	**260.67**	3.41
	60	342.53	**0.10**	344.53	**0.10**	**347.40**	7.06
	80	442.27	**0.20**	**444.40**	0.30	444.33	11.80
	100	529.73	**0.40**	530.53	0.70	**545.20**	15.92
$p = 4$	120	574.53	**0.70**	577.00	1.00	**594.07**	20.29
	140	595.47	**1.50**	598.47	2.00	**629.20**	25.98
	160	628.27	**2.00**	625.00	2.70	**663.67**	31.30
	180	734.27	**2.40**	732.20	3.80	**780.87**	39.68
	200	836.20	**3.80**	835.93	5.80	**873.73**	48.08
	40	253.07	**0.10**	251.07	**0.10**	**258.40**	4.16
	60	330.87	**0.10**	332.00	**0.10**	**333.20**	8.42
	80	425.80	**0.20**	**430.27**	0.30	426.47	14.06
	100	524.80	**0.50**	517.20	0.90	**530.53**	18.74
$p = 5$	120	562.33	**0.90**	556.67	1.40	**579.20**	24.02
	140	585.80	**1.80**	583.40	2.00	**609.20**	30.72
	160	617.33	**2.50**	609.33	2.90	**637.60**	36.73
	180	712.33	**3.20**	712.80	4.40	**740.60**	46.07
	200	816.40	**5.20**	818.13	6.90	**841.33**	56.00

Table 4.4: Comparison of I1 and I2 interchange heuristics of [2] with GA on instances with $T = 0.1$, $U_{\%} = 0.85$

		I1		I2		GA	
	n	Obj.	Time	Obj.	Time	Obj.	Time
	40	175.00	**0.00**	**180.80**	**0.00**	177.60	2.41
	60	175.80	**0.00**	**176.87**	**0.00**	171.73	4.84
	80	240.13	**0.10**	**240.33**	**0.10**	239.07	8.04
	100	244.07	**0.10**	247.07	0.30	**254.73**	11.23
$p = 3$	120	258.60	**0.20**	256.93	0.40	**273.13**	13.94
	140	213.67	**0.30**	223.00	0.70	**244.33**	17.54
	160	223.07	**0.50**	228.80	0.70	**264.87**	21.10
	180	287.93	**0.70**	291.00	1.40	**328.87**	26.31
	200	300.47	**0.90**	301.73	1.70	**344.13**	32.60
	40	168.53	**0.00**	**174.27**	**0.00**	173.60	2.99
	60	165.00	**0.00**	**167.40**	0.10	164.80	5.96
	80	224.53	**0.10**	**227.60**	**0.10**	226.27	9.69
	100	235.53	**0.20**	238.27	0.30	**245.87**	13.26
$p = 4$	120	238.60	**0.20**	239.40	0.40	**254.93**	17.11
	140	207.13	**0.50**	212.07	0.90	**231.00**	21.44
	160	211.13	**0.60**	213.53	0.90	**229.20**	25.50
	180	280.53	**1.10**	284.53	1.90	**309.00**	31.72
	200	290.80	**1.50**	292.53	2.30	**313.00**	38.67
	40	164.60	**0.00**	**168.80**	**0.00**	167.87	3.57
	60	157.13	**0.00**	160.47	0.10	**160.80**	7.08
	80	217.00	**0.10**	**220.33**	0.20	217.47	11.45
	100	222.00	**0.20**	223.60	0.40	**232.53**	15.74
$p = 5$	120	227.00	**0.40**	221.73	0.50	**236.67**	20.23
	140	198.93	**0.50**	200.33	0.90	**218.53**	25.26
	160	198.73	**0.70**	204.33	0.90	**223.27**	30.50
	180	266.07	**1.30**	264.80	1.60	**295.47**	37.24
	200	276.80	**2.00**	276.53	2.70	**303.60**	45.69

Table 4.5: Comparison of I1 and I2 interchange heuristics of [2] with GA on instances with $T = 0.3$, $U_{\%} = 0.65$

		I1		I2		GA	
	n	Obj.	Time	Obj.	Time	Obj.	Time
	40	**355.27**	**0.60**	355.13	0.70	354.87	3.16
	60	**540.80**	**0.80**	540.20	1.30	540.73	6.99
	80	**718.87**	**1.70**	**718.87**	2.50	717.47	12.10
	100	851.20	**2.70**	851.20	4.60	**877.93**	16.79
$p = 3$	120	1014.00	**3.40**	1011.80	7.30	**1040.07**	23.99
	140	1132.60	**5.50**	1130.73	9.80	**1209.53**	31.72
	160	1234.67	**6.60**	1250.00	9.50	**1331.20**	40.78
	180	1483.47	**8.20**	1483.07	11.40	**1559.33**	52.82
	200	1638.33	**15.90**	1635.73	23.80	**1728.73**	64.87
	40	348.33	**0.80**	348.13	0.90	**348.53**	4.02
	60	520.27	**1.50**	520.33	1.70	**521.33**	8.81
	80	699.33	**1.40**	**699.67**	2.80	699.53	15.12
	100	843.47	**3.60**	843.47	8.80	**850.67**	20.72
$p = 4$	120	990.73	**5.40**	988.00	8.80	**1014.53**	29.69
	140	1115.53	**4.90**	1112.40	10.90	**1164.87**	40.02
	160	1207.00	**6.10**	1226.00	10.80	**1287.27**	50.33
	180	1436.00	**7.90**	1434.53	18.60	**1506.20**	64.55
	200	1588.40	**12.90**	1588.87	26.50	**1664.60**	79.65
	40	342.20	**1.60**	342.07	1.90	**343.20**	4.88
	60	511.47	**0.90**	513.20	2.10	**514.07**	10.60
	80	**684.07**	**1.80**	683.07	3.00	681.40	18.21
	100	820.07	**8.00**	819.67	13.10	**837.60**	24.98
$p = 5$	120	969.07	**3.80**	967.33	11.40	**992.53**	35.83
	140	1095.60	**5.90**	1091.27	11.10	**1134.33**	47.48
	160	1182.13	**8.30**	1199.87	13.20	**1249.73**	59.43
	180	1431.27	**9.40**	1427.13	15.40	**1470.07**	76.65
	200	1572.53	**27.50**	1572.13	32.70	**1631.53**	92.84

Table 4.6: Comparison of I1 and I2 interchange heuristics of [2] with GA on instances with $T = 0.3$, $U_{\%} = 0.75$

		I1		I2		GA	
	n	Obj.	Time	Obj.	Time	Obj.	Time
	40	306.67	**0.20**	307.73	0.30	**308.13**	3.00
	60	438.27	**0.20**	437.33	0.40	**438.73**	6.45
	80	575.87	**0.40**	**585.07**	0.60	577.07	11.09
	100	639.53	**0.40**	640.93	0.90	**686.40**	15.25
$p = 3$	120	743.20	**1.00**	740.33	1.90	**806.53**	20.97
	140	754.33	**2.10**	765.20	2.60	**901.07**	27.76
	160	813.00	**2.50**	808.93	3.20	**955.00**	34.71
	180	941.80	**3.50**	937.80	4.70	**1153.47**	44.89
	200	1092.40	**4.90**	1092.47	7.10	**1281.93**	54.36
	40	288.53	**0.20**	288.40	0.30	**290.73**	3.72
	60	404.80	**0.20**	404.40	0.40	**409.67**	7.98
	80	530.73	**0.30**	**540.00**	0.70	531.73	13.34
	100	613.67	**0.60**	615.33	1.10	**636.80**	18.36
$p = 4$	120	689.93	**1.10**	685.00	2.10	**733.40**	24.63
	140	706.93	**2.80**	720.27	3.70	**813.07**	32.54
	160	764.40	**2.80**	764.13	4.30	**872.60**	39.78
	180	890.80	**5.10**	887.93	6.70	**1017.47**	51.13
	200	1026.87	**8.50**	1024.80	12.80	**1153.07**	61.92
	40	277.60	**0.40**	280.73	0.60	**281.80**	4.59
	60	393.27	**0.30**	393.27	0.40	**395.87**	9.50
	80	497.60	**0.40**	501.67	0.90	**504.87**	15.64
	100	593.80	**0.70**	595.47	1.50	**615.93**	21.30
$p = 5$	120	671.80	**1.50**	666.27	3.00	**692.20**	28.46
	140	686.40	**3.10**	697.13	4.50	**757.60**	36.98
	160	728.67	**4.70**	729.47	5.80	**796.20**	45.39
	180	857.93	**6.20**	854.47	7.10	**942.20**	58.11
	200	973.00	**8.10**	971.07	9.80	**1096.00**	71.02

Table 4.7: Comparison of I1 and I2 interchange heuristics of [2] with GA on instances with $T = 0.3$, $U_{\%} = 0.85$

		I1		I2		GA	
	n	Obj.	Time	Obj.	Time	Obj.	Time
	40	229.60	**0.00**	233.60	0.10	**234.67**	2.80
	60	273.67	**0.10**	273.73	**0.10**	**276.40**	5.78
	80	359.33	**0.10**	**366.13**	0.20	358.07	9.84
	100	358.27	**0.30**	360.13	0.40	**427.93**	13.29
$p = 3$	120	382.13	**0.50**	374.87	0.60	**456.53**	17.08
	140	320.93	**0.90**	324.87	1.30	**468.33**	22.05
	160	336.80	**0.80**	349.47	1.40	**487.80**	27.14
	180	416.93	**1.70**	416.67	2.10	**580.00**	34.53
	200	464.73	**2.30**	471.73	3.30	**638.00**	42.77
	40	207.13	**0.00**	**212.20**	0.10	210.07	3.41
	60	235.47	**0.10**	228.80	**0.10**	**235.93**	6.92
	80	**303.07**	**0.20**	301.27	0.30	302.33	11.42
	100	323.27	**0.30**	323.67	0.40	**358.33**	15.28
$p = 4$	120	330.53	**0.70**	330.67	**0.70**	**374.80**	19.39
	140	289.67	**0.90**	290.40	1.20	**371.60**	24.58
	160	304.07	**1.20**	309.60	1.40	**385.87**	29.94
	180	382.20	**1.80**	382.40	2.50	**463.67**	37.89
	200	402.73	**2.60**	407.53	3.70	**526.80**	45.06
	40	192.60	**0.00**	194.53	0.10	**200.20**	4.10
	60	**223.87**	**0.10**	214.53	**0.10**	221.13	8.02
	80	279.07	**0.20**	**279.40**	0.30	277.73	13.18
	100	304.33	**0.40**	303.27	0.60	**322.87**	17.91
$p = 5$	120	300.33	**0.60**	302.27	0.90	**328.60**	22.61
	140	276.67	**1.00**	277.47	1.50	**329.13**	28.60
	160	280.20	**1.30**	293.00	1.70	**343.40**	33.77
	180	365.60	**2.80**	363.00	2.90	**411.80**	41.76
	200	381.20	**3.20**	386.67	4.40	**466.20**	51.14

Table 4.8: Comparison of I1 and I2 interchange heuristics of [2] with GA on instances with $T = 0.5$, $U_{\%} = 0.7$

		I1		I2		GA	
	n	Obj.	Time	Obj.	Time	Obj.	Time
	40	360.07	**0.70**	**360.33**	0.80	359.07	3.30
	60	552.40	**1.20**	**552.53**	1.40	551.40	7.17
	80	**733.53**	**2.40**	733.47	2.80	733.47	12.41
	100	845.27	**8.60**	844.07	6.80	**895.07**	17.27
$p = 3$	120	996.07	**5.80**	995.07	7.60	**1059.93**	24.65
	140	1078.53	**6.70**	1084.00	10.80	**1232.20**	32.79
	160	1168.27	**7.70**	1167.80	11.30	**1365.33**	42.86
	180	1384.80	**14.20**	1392.60	25.60	**1593.53**	54.35
	200	1548.27	**29.60**	1548.27	42.70	**1760.27**	67.85
	40	346.27	**0.90**	345.60	1.90	**346.73**	4.18
	60	520.27	**1.20**	**520.47**	1.90	520.27	8.94
	80	686.60	**4.20**	**688.73**	3.90	688.60	15.42
	100	800.40	**4.10**	802.73	8.40	**843.60**	21.37
$p = 4$	120	941.67	**6.50**	947.87	8.10	**991.00**	29.97
	140	1038.73	**6.70**	1041.60	12.30	**1145.93**	40.35
	160	1102.27	**6.90**	1104.13	12.30	**1258.73**	51.60
	180	1326.47	**18.50**	1334.93	27.70	**1474.47**	65.59
	200	1497.20	**30.80**	1496.93	33.70	**1638.40**	81.27
	40	**339.40**	**1.10**	333.93	1.50	338.27	5.01
	60	**498.40**	**2.00**	498.20	**2.00**	497.60	10.68
	80	662.67	**2.30**	653.60	3.90	**664.00**	17.99
	100	775.00	**3.50**	774.67	6.70	**798.53**	25.05
$p = 5$	120	905.80	**5.80**	914.00	10.50	**944.67**	35.19
	140	1006.93	**8.20**	1014.33	11.40	**1081.00**	46.32
	160	1069.87	**8.00**	1069.53	13.40	**1176.20**	58.74
	180	1279.13	**14.70**	1296.47	15.50	**1392.60**	75.60
	200	1443.53	**25.80**	1439.33	34.60	**1539.87**	93.22

Table 4.9: Comparison of I1 and I2 interchange heuristics of [2] with GA on instances with $T = 0.5$, $U_{\%} = 0.8$

		I1		I2		GA	
	n	Obj.	Time	Obj.	Time	Obj.	Time
	40	**318.27**	**0.20**	317.87	0.30	317.27	3.13
	60	**460.20**	**0.40**	**460.20**	0.60	450.73	6.81
	80	**604.07**	**0.60**	**604.07**	0.80	600.73	11.59
	100	630.07	**0.60**	629.93	1.10	**705.13**	16.08
$p = 3$	120	720.40	**1.50**	726.00	1.90	**840.33**	22.67
	140	681.67	**2.50**	677.40	3.10	**942.93**	29.94
	160	702.93	**2.30**	708.13	3.50	**1002.13**	37.92
	180	848.60	**4.80**	857.47	6.30	**1201.13**	48.69
	200	962.07	**5.00**	963.53	8.40	**1337.80**	60.62
	40	290.33	**0.30**	**291.20**	**0.30**	290.80	3.88
	60	**405.80**	**0.30**	404.20	1.40	401.13	8.09
	80	530.13	**0.50**	**531.33**	0.90	520.73	13.76
	100	565.00	**0.60**	567.80	1.00	**622.47**	18.60
$p = 4$	120	627.33	**1.50**	638.47	2.30	**718.20**	25.87
	140	616.47	**2.40**	619.93	3.20	**782.00**	33.68
	160	636.87	**2.80**	640.27	4.30	**832.67**	41.37
	180	756.80	**4.90**	768.33	5.80	**972.40**	52.88
	200	859.67	**5.60**	858.67	10.70	**1108.07**	65.44
	40	267.53	**0.20**	**271.20**	0.30	270.67	4.56
	60	366.13	**0.20**	365.47	0.60	**368.33**	9.45
	80	469.20	**0.80**	470.40	1.10	**472.20**	15.72
	100	536.67	**0.90**	536.93	1.30	**566.93**	21.11
$p = 5$	120	581.20	**3.10**	585.27	4.40	**639.47**	28.59
	140	578.07	**2.50**	579.27	4.00	**680.13**	36.90
	160	587.87	**3.50**	594.13	4.60	**719.00**	45.08
	180	695.07	**4.70**	700.00	6.70	**856.27**	57.88
	200	760.20	**8.20**	783.53	10.70	**968.33**	70.75

Table 4.10: Comparison of I1 and I2 interchange heuristics of [2] with GA on instances with $T = 0.5, U_{\%} = 0.9$

		I1		I2		GA	
	n	Obj.	Time	Obj.	Time	Obj.	Time
	40	**252.33**	**0.10**	251.07	**0.10**	243.60	2.93
	60	307.33	**0.10**	**312.33**	0.20	305.67	6.18
	80	**416.40**	**0.20**	412.67	0.30	402.73	10.55
	100	351.60	**0.30**	356.47	0.50	**465.87**	14.21
$p = 3$	120	378.87	**0.70**	371.00	1.00	**512.40**	19.47
	140	287.87	**1.00**	287.60	1.20	**541.53**	25.37
	160	299.67	**1.20**	307.20	1.50	**560.47**	31.30
	180	366.60	**1.50**	374.27	2.30	**655.00**	41.16
	200	385.13	**2.50**	396.47	4.00	**739.87**	49.84
	40	**211.47**	**0.00**	211.27	0.10	207.73	3.53
	60	230.87	**0.10**	**236.00**	**0.10**	234.53	7.10
	80	**303.47**	**0.20**	300.33	0.40	301.53	11.87
	100	292.00	**0.50**	291.53	0.70	**353.13**	16.20
$p = 4$	120	297.13	**0.70**	290.33	0.90	**381.47**	20.68
	140	232.67	**1.00**	237.33	1.30	**370.73**	27.02
	160	246.07	**0.90**	255.73	1.50	**395.27**	32.51
	180	304.67	**1.80**	315.67	2.50	**455.53**	41.46
	200	324.13	**2.30**	326.33	4.00	**523.40**	50.46
	40	**184.80**	**0.10**	184.73	**0.10**	182.33	4.08
	60	184.27	**0.10**	194.53	**0.10**	**198.47**	8.07
	80	**266.07**	**0.20**	264.20	0.30	258.00	13.36
	100	250.53	**0.50**	248.27	0.80	**290.80**	18.39
$p = 5$	120	256.67	**0.50**	250.73	0.90	**308.07**	23.22
	140	205.07	**0.90**	208.47	1.30	**295.00**	29.48
	160	208.07	**1.40**	217.73	1.80	**306.93**	36.63
	180	266.33	**2.00**	284.27	2.70	**370.67**	44.17
	200	286.27	**3.20**	291.33	4.10	**396.47**	54.09

Our approach has performed better than I1 and I2 on all the instances when the number of nodes is 100 and above. For the smaller datasets when the number of nodes is 40, 60 and 80, only in some cases our results are worse than I1 or I2 results. Coming to the execution

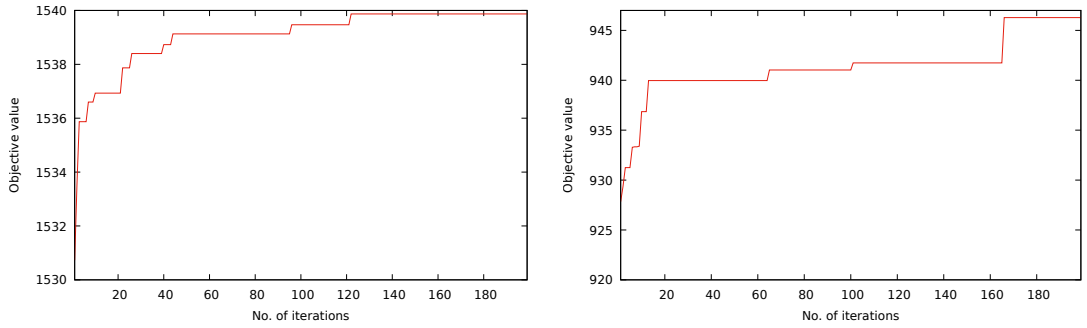
times, Averbakh et al. [2] have run the interchange heuristics, I1 and I2, on a AMD Phenom II processor with 3.31 GHz and 16 GB of RAM. Our approach is executed on a different system with lower RAM size of 8GB, so we can not directly compare the execution times. But it is clear that I1 and I2 are faster than our approach. This is also expected as our approach being a population based metaheuristic approach will require longer execution times.

Table 4.11 presents the comparison of our results with I1 and I2 in the same format as used in [2]. In Table 4.11, the first column gives the threshold value T and the second column gives the fraction of the network diameter $U_{\%}$. The third and fourth columns give the average percentage improvement in solution quality by GA with respect to I1 and I2 respectively over all the instances with the same T and $U_{\%}$ values. Given a particular instance and any two methods P, Q which are executed on that instance and obtained solutions X_P, X_Q respectively, we calculate the percentage improvement in solution quality by method P over method Q on that instance as $100 \times \frac{\phi_i(X_P)-\phi_i(X_Q)}{\phi_i(X_Q)}$. The average percentage improvements presented in columns 3 and 4 are averaged over 405 instances which have the same T and $U_{\%}$ values. The fifth, sixth and seventh columns in the Table 4.11 give the average execution times over all the 45 instances which have the maximum number of nodes, i.e. $n = 200$, for I1, I2 and GA respectively. This table, which we have provided with the sole purpose of comparing the results in the same format as in [2], also shows the superiority of GA over I1 and I2 in terms of solution quality.

Table 4.11: Comparison of I1 and I2 interchange heuristics of [2] with GA in the same format as in [2]

T	$U_{\%}$	%(GA, I1)	%(GA, I2)	R(I1)	R(I2)	R(GA)
0.1	0.65	1.97	1.34	8.3	12.9	68.4
0.1	0.75	4.01	3.85	3.9	5.5	48.5
0.1	0.85	8.66	5.88	1.5	2.3	38.9
0.3	0.65	3.01	2.88	18.8	27.7	79.1
0.3	0.75	9.30	8.75	7.2	9.9	62.4
0.3	0.85	22.19	20.89	2.7	3.8	46.3
0.5	0.7	6.67	6.56	28.8	37.0	80.8
0.5	0.8	19.12	18.13	6.3	10.0	65.6
0.5	0.9	49.41	46.58	2.7	4.0	51.5

To show the convergence behaviour of our GA, we have taken two classes of instances. Fig. 4.2a and Fig. 4.2b show the convergence behavior of our GA on instances with $\{n = 200,$



(a) Instances with $n = 200$, $p = 5$, $T = 0.5$, $U_{\%} = 0.7$

(b) Instances with $n = 200$, $p = 5$, $T = 0.5$, $U_{\%} = 0.8$

Figure 4.2: Convergence behaviour of GA

$p = 5, T = 0.5, U_{\%} = 0.7\}$ and $\{n = 200, p = 5, T = 0.5, U_{\%} = 0.8\}$ respectively. In these two figures, objective value represents the average objective value over 15 instances belonging to the respective class. We have allowed our GA to execute for 200 iterations only. These figures show that our GA converges rapidly to high quality solutions within 200 iterations.

Table 4.12: Wilcoxon Signed-Ranks test of GA with I2

N	NWT	W^+	W^-	z	z_c	Significant
3645	2374	153044	2666081	-37.62	-2.58	Yes

We have conducted two non-parametric statistical tests, viz. Wilcoxon-signed rank test [84] and Friedman test [126, 127] to check the significance of the results obtained by our approach. We have performed the two-tailed Wilcoxon-signed rank test by setting the significance criteria to 1% (i.e. p-value ≤ 0.01) between our GA based approach and I2 which is the better of the two interchange heuristics proposed in [2]. Table 4.12 presents the outcome of this test. In this table, the first column N represents the total number of instances considered. The second column, NWT is the number of instances without a tie. The third column, W^+ is the sum of ranks of the instances where I2 performed better than GA, whereas the fourth column W^- gives the sum of ranks of the instances where GA performed better than I2. The fifth column gives the z value obtained, -37.62 which is less than the critical value of z, $z_c = -2.58$ presented in the sixth column of the table. This shows the better performance of GA over I2 is statistically significant as presented in the last column of the Table 4.12.

Table 4.13 corresponds to Friedman test. In this table, the first column N represents the total number of instances considered. The second column df gives the degrees of freedom which

Table 4.13: Friedman test of I1, I2 and GA

N	df	α	χ_c^2	χ^2	Significant
3645	2	0.01	9.21	1691.64	Yes

Table 4.14: Mean ranks for I1, I2 and GA in Friedman test

I_1	I_2	GA
1.7	1.7	2.6

is equal to the number of groups in the data minus 1. The third column gives the $alpha$ value considered which is 0.01, i.e., significance criteria is set to 1%. The fourth column is the critical value of the χ_c^2 for the present values of $alpha$ and df. The fifth column χ^2 is the test statistic calculated from the data. The value of $\chi^2 = 1691.64$ is greater than the critical value of the $\chi_c^2 = 9.21$, which proves that there is significant difference in the performance of the considered approaches. Table 4.14 presents the mean ranks for I1, I2 and GA in the first column, the second column and the third column respectively. The mean of the ranks for GA approach is greater than the means of the ranks for I1 and I2, thereby proving the better performance of GA as compared to I1 and I2 according to the Friedman test.

All these results show that the performance of our GA approach is better in terms of solution quality in comparison to the two interchange heuristics, and non-parametric statistical tests results of the Wilcoxon signed-rank test and the Friedman test also prove that this superior performance of our approach is due to algorithmic merit.

4.5 Conclusions

In this chapter, we have proposed an evolutionary approach, viz. GA based approach for the OCMCLP. The results obtained with our approach are compared with two interchange heuristics available in the literature. On most of the instances, our GA based approach has found superior quality solutions in comparison to the existing methods. However, our approach needs more execution time than these methods. Our GA based approach is the maiden metaheuristic approach that has been developed for OCMCLP.

Chapter 5

Reliability p-median problem

5.1 Introduction

Facility location problems deal with identifying suitable locations for a set of facilities that serve a set of demand points or customers. Among the facility location problems, p-median problem (pMP) is a well studied problem which was introduced by Hakimi [128] in 1964. The pMP is concerned with locating p facilities in such a way that the sum total of demand-weighted distances between each demand point and its respective closest facility is minimized. It is an $\mathcal{NP}$-hard problem as shown by Kariv and Hakimi [129]. Starting from the first formulation provided in [130], there have been several different formulations of the pMP over the years as mentioned in [131, 132, 133]. The capacitated p-median problem (CPMP) is a variant of pMP in which there is a capacity constraint on each of the possible facilities and each facility can only serve demand points within its capacity limit. The traditional pMP is an uncapacitated version where there is no capacity constraint on the facilities. Over the last few decades, several authors have proposed different methods to solve the pMP and its variants. Jayalakshmi and Singh [44] proposed a swarm intelligence approach using artificial bee colony algorithm to solve the pMP. In [132] Domínguez and Muñoz introduced a new reduced formulation for the pMP and proposed a recurrent neural network to solve this new formulation. In [133], Sourour proposed a tighter formulation of the pMP by mixed integer linear program and demonstrated that the standard branch-and-cut algorithm efficiently solves this tighter formulation on the considered benchmark instances. Keivan and Seyed [134] proposed a genetic algorithm based approach for the CPMP. Fleszar and Hindi [135] proposed a variable neighbourhood search based solution for the CPMP. In [136], Canós et al. introduced fuzzy p-median problem that

allows some of the demand points to be uncovered if that gives significant lower cost. The fuzzy p-median problem finds its applications in private sector firms that try to maximize the profit while providing non-essential services and need not serve all the demand points. Canós et al. [136] also proposed an exact algorithm to solve this problem. In [137], Cadenas et al. proposed a two-population genetic algorithm to solve the fuzzy p-median problem. The vector assignment p-median problem (VAPMP) [138] and distributed p-median problem [139] are the other variants of pMP in which the demand of a customer is collectively satisfied by different facilities at various levels and not just the closest facility.

In the p-median problem, it is assumed that once constructed, the facilities will always operate as planned. But in reality, facilities may fail at times due to several reasons like natural disasters such as floods and earthquakes. Facilities may also fail due to events which are intentional like terrorist attacks and labor strikes. Such intentional activity disrupting a system is called interdiction [140]. Sometimes facilities may fail due to unintended events like sudden power or component failures. When there are facility failures, there will be disruptions in the services provided to the customers. Depending on the type and severity of the events that make the facilities inoperable, the disruptions to the customer service can last for a short duration or for a longer period of time. In such facility failure cases, a customer who is generally serviced by the nearest facility now needs to be assigned to a distant functional facility. This increases the overall cost due to the additional distance the customer has to travel to a new facility to avail the services. For the first time, Snyder and Daskin [141] proposed the reliability p-median problem which introduced reliability approach in the facility location model taking facility failures into consideration. The reliability p-median problem (RpMP) minimizes both the primary transportation cost without considering the facility failures and also the cost of the expected failure considering the facility failures. The basic idea behind introducing the RpMP was to allot each customer to a primary facility which will serve it in normal situations, as well as to a set of backup facilities which will serve it in the case of failure of the primary facility. As multiple failures can happen at the same time, each customer requires a first backup facility when its primary facility fails, a second backup facility when its first backup facility fails, and so on. However, if a customer is allotted to a nonfailable facility as its k^{th} backup, no more backups are required. Usually, nonfailable facilities are much less in numbers in comparison to failable ones owing to higher cost of the former. If all facilities are nonfailable then RpMP reduces to p-median problem. Owing to the provision of backup facilities at multiple levels,

RpMP is a highly complex problem which can be used for modelling critical applications in different domains such as health care, aviation and defence [141].

Snyder and Daskin [141] while introducing RpMP also proposed an exact method for solving it based on Lagrangean relaxation. This method was suitable for solving small instances only. Later Alcaraz et al. [142] proposed eight hybrid metaheuristic approaches for RpMP based on genetic algorithm (GA) and scatter search (SS). Among these eight hybrid approaches, four (GA1, GA2, GA3, GA4) were based on GA and four (SS1, SS2, SS3, SS4) were based on SS. Among these eight approaches, GA2 and SS2 performed better than other approaches with SS2 performing slightly better than GA2. These are the only approaches available in the literature for RpMP. However, several problems which consider reliability issues in various kind of networks have been studied in the literature in the last few decades.

Several authors studied reliability of telecommunication networks and power networks considering the failure of edges connecting the nodes [143, 144, 145, 146]. These connectivity reliability theories deal with the probability of the network being connected even when there are failures along edges. There are several approaches proposed over the years to estimate the network reliability considering failures along the edges [147, 148, 149, 150]. Not only the edge failures, taking the facility disruptions also into consideration, there have been several approaches proposed in the past to deal with network reliability issues [141, 151, 152, 153, 154, 155, 156]. In [141], an equal uniform probability of failure was considered for all the candidate locations. There are also other models which do not consider the equal uniform probability of failure for all the candidate locations [152, 157]. In [152], it is shown that in the case of high probability of failure, the facilities are centrally located or even co-located in some cases. On the other hand, as the probability of failure decreases and approaching zero, in order to reduce the travel costs, the facilities typically disperse until they reach the optimal locations suitable for the problem not considering facility failures. Based on the assumptions made, several variants of the problem are possible. In [158, 159] the authors have studied interdependent facility failures as compared to the independent facility failures. Likewise, in [160, 161] the authors have studied capacitated reliable facility location problems. Lim et al. [162] studied the case of hardening certain facilities by making them non-failable with an additional cost while the other facilities are prone to random facility disruptions. In [154] a mixed-integer programming model was presented with an objective of minimizing operational cost when there are no disruptions and also reducing the risk of disruption by applying the p-robustness criterion. The p-robustness criterion was used to limit the cost in the case of disruption. Similarly, [163] dealt with a facility location problem

considering the uncertainty regarding future events. The uncertainty of the future events was modeled by providing different future situations with associated probabilities. A model known as the α-reliable mean-excess model was introduced in [163] that aims to minimize the expected regret from a list of worst-case scenarios which will occur with a probability of $1 - \alpha$. Some facilities may be highly vulnerable to interdiction due to their geographical locations causing significant impact in the case of unavailability of such facilities. The interdiction models deal with identifying most critical failures and take them into consideration [140, 164, 165]. [166] deals with fortifying some of the facilities within a finite budget.

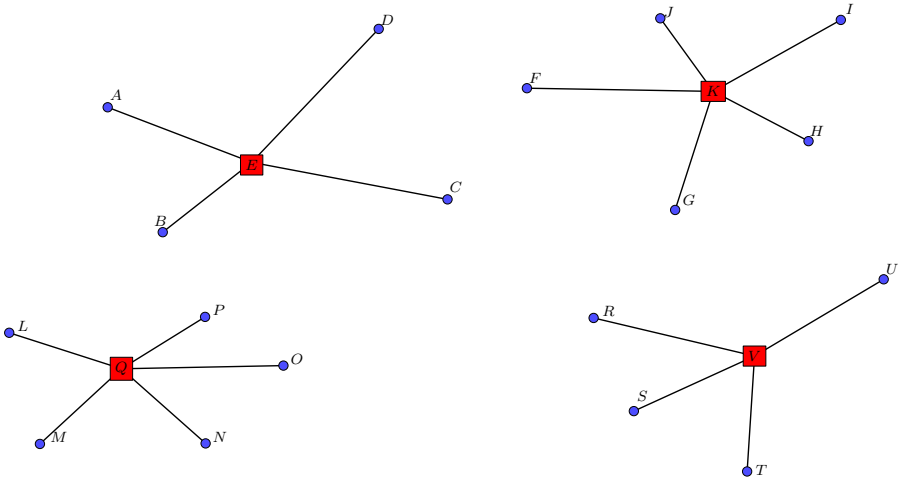
Unlike the vector assignment p-median problem (VAPMP), in RpMP a customer is served by a level 2 facility only when the previous facility at level 1 fails. There have been location models introduced based on queueing of customers waiting to receive service from a facility in congested scenarios [167, 168]. Consolidating several methods of solving the facility location problems in systems with congestion, Berman and Krass [169] presented a complex model in an illustrative manner. RpMP differs from the facility location in congested systems as we deal with complete failure of a facility in RpMP as compared to a facility being unable to serve customers due to congestion. Afify et al. [170] and Afify et al. [171] studied the related facility location problems where there is only one layer of backup and limited budget is available to fortify a few facilities to make them nonfailable. In comparison to these problems, the model of RpMP introduced in [141] is way more complex and more relevant for critical applications.

Consider an example network with 22 nodes and 4 facilities as depicted in Figure 5.1. The nodes selected for facilities are shown in rectangle shape and marked in red color for easy identification. As shown in the sub-figure 5.1a, nodes $\{A, B, C, D\}$ are assigned to their nearest facility E. Nodes $\{F, G, H, I, J\}$ are assigned to facility K. Similarly nodes $\{L, M, N, O, P\}$ are assigned to facility Q and finally nodes $\{R, S, T, U\}$ are assigned to facility V. When the facility E fails, out of the nodes assigned to it, $\{C, D\}$ are now re-assigned to the next closest functioning facility which is K as shown in sub-figure 5.1b. And the nodes $\{A, B\}$ including the facility E is now assigned to facility Q which is the closest functioning facility. The re-assignment of nodes $\{A, B, C, D, E\}$ to the new facilities is depicted in dotted lines in sub-figure 5.1b.

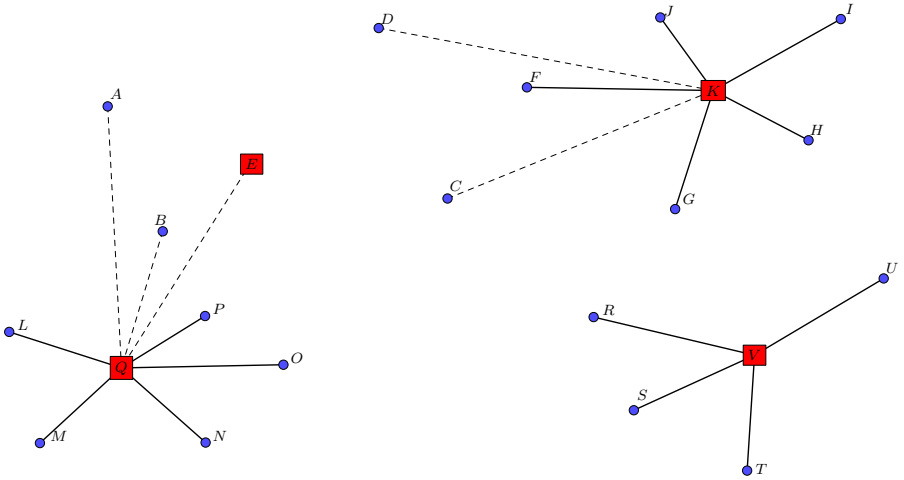
In this chapter, we have proposed a hyper-heuristic based approach with naive Bayes classifier for solving the reliability p-median problem (RpMP). We have compared the results of our approach with the state-of-the-art methods available in the literature [142]. The effectiveness of the proposed approach can be observed from the superior quality of our solutions. Further,

use of naive Bayes classifier not only reduces the total execution time but also improves the solution quality.

The remainder of this chapter is organized as follows: Section 5.2 presents the formal problem definition of RpMP, Section 5.3 explains functioning of naive Bayes with an example, Section 5.4 presents the proposed greedy selection based hyper-heuristic approach for RpMP. Experimental results and their analysis are presented in Section 5.5. Finally, Section 5.6 concludes the chapter by providing a summary of contributions made.



(a) Node assignments when all the four facilities are functioning



(b) Node assignments when one facility fails

Figure 5.1: Illustration of reliability p-median problem with $p = 4$ facilities

Table 5.1: Summary of key notations

Notation	Meaning
A	Set of demand points or customers
B	Set of potential locations for facilities
F	Subset of locations of candidate facilities(B) which may fail
NF	Subset of locations of candidate facilities(B) which may not fail
h_i	Demand at each customer $i \in A$
d_{ij}	Cost of service per unit of demand of customer $i \in A$ from facility $j \in B$
θ_i	Cost per unit of demand of customer $i \in A$ when i remains unserved
q	Uniform probability of failure of each facility $j \in F$
p	Total number of facilities to be located

5.2 Formal problem definition

To formally define RpMP, we have followed the notational conventions of [142]. Table 5.1 gives the summary of key notational conventions.

In the RpMP modeling, each customer $i \in A$ is either assigned to a failable facility or to a non-failable facility. A non-failable facility will never fail, and there is no need to consider re-assigning customer i to any other facility. But if the customer i is assigned to a failable facility then it is assigned to different facilities at different levels, viz. to an open facility at level 0, to a different facility at level 1 which will serve the customer i if the facility at level 0 fails, and i is assigned to yet another facility at level 2 when both the facilities at level 0 & level 1 fail, and so on till all the open facilities are considered. To represent the open facilities and the assignment of a customer to a facility at a level r, where $r = 0, \ldots, p-1$, we use two binary variables, viz. $S_j \in \{0,1\}$, $W_{ijr} \in \{0,1\}$. S_j, W_{ijr} are referred to as location binary variable and assignment binary variable respectively. As the name suggests, $S_j = 1$ if there is an open facility at $j \in B$, otherwise $S_j = 0$. $W_{ijr} = 1$ if customer $i \in A$ is assigned to facility $j \in B$ at level r, otherwise $W_{ijr} = 0$.

Considering $\alpha \in [0,1]$ and $R = \{0, 1, ..., p-1\}$, RpMP can be mathematically formulated [141] in the following manner:

$$(RpMP) \;\; minimize \;\; \alpha c_1 + (1-\alpha)\, c_2 \tag{5.1}$$

subject to

$$\sum_{j \in B} W_{ijr} + \sum_{j \in NF} \sum_{x=0}^{r-1} W_{ijx} = 1 \;\; \forall i \in A, r \in R \tag{5.2}$$

$$W_{ijr} \leq S_j \;\; \forall i \in A, j \in B, r \in R \tag{5.3}$$

$$\sum_{j \in B} S_j = p \tag{5.4}$$

$$\sum_{r \in R} W_{ijr} \leq 1 \;\; \forall i \in A, j \in B \tag{5.5}$$

$$S_u = 1 \tag{5.6}$$

$$S_j \in \{0, 1\} \;\; \forall j \in B \tag{5.7}$$

$$W_{ijr} \in \{0, 1\} \;\; \forall i \in A, j \in B, r \in R \tag{5.8}$$

where

$$c_1 = \sum_{i \in A} \sum_{j \in B} h_i d_{ij} W_{ij0} \tag{5.9}$$

$$c_2 = \sum_{i \in A} h_i \left[\sum_{j \in NF} \sum_{r \in R} d_{ij} q^r W_{ijr} + \sum_{j \in F} \sum_{r \in R} d_{ij} q^r \left(1 - q\right) W_{ijr} \right] \tag{5.10}$$

Equation 5.1 gives the objective function of the problem which is the weighted sum of c_1 which is primary cost of serving customers assuming none of the facilities fail, and c_2 which is the expected failure cost considering the facility failures. Constraint in equation 5.2 makes sure that given a customer i and level r, either i is served by a facility $j \in B$ at level r or i is served by a non-failable facility $j \in NF$ at level $x < r$. Constraint in equation 5.3 ensures that a customer i can only be assigned to a location where a facility is located. Constraint in equation 5.4 requires that there are exactly p facilities located. Constraint in equation 5.5 requires that a customer i is assigned to a facility at one level at most. Constraint in equation 5.6 gives the information of an emergency facility being opened at an imaginary location u which is non-failable, $u \in NF$. For each customer $i \in A$, if it remains unserved because it is not connected to any of the facilities or

all the facilities have failed then it is considered that the customer i is served by the emergency non-failable facility u with a transportation cost of $d_{iu} = \theta_i$. Equations 5.7,5.8 give the binary nature of the location variables and assignment variables. Using equation 5.9, the primary cost of serving customers assuming none of the facilities fail c_1 is calculated, whereas using equation 5.10 the expected failure cost c_2 considering the facility failures is calculated.

5.3 Naive Bayes classifier

Naive Bayes classifier is a supervised learning approach. It is a probabilistic model and works according to the Bayes theorem. In the naive Bayes classifier, it is assumed that the features which are involved in classification are independent of each other given the class variable. Naive Bayes is a simple yet computationally efficient classifier which is unaffected by noise. Over the years, naive Bayes classifier has been extensively used for text classification [172, 173], in traffic risk management [174], to predict Alzheimer's disease from genome-wide data [175]. But no one so far applied naive Bayes to solve combinatorial optimization problems so as to improve the solution quality or reduce the execution time of the approaches. This served as a motivation for us to incorporate naive Bayes in our proposed hyper-heuristic approach.

Every classifier has two major phases namely training phase and testing phase [176]. As part of the training phase, a prediction model is generated which correlates different features to a class label. In the naive Bayes classifier, at the end of the training phase, we calculate the class probability of each of the classes and we also calculate the conditional probabilities for each of the feature values given a class. We calculate the conditional probability of each feature value as ratio of the number of occurrences of that feature value for a given class to the total number of instances belonging to that class.

Probability of a feature vector s belonging to a particular class Y_j is calculated using Bayes theorem as follows:

$$p(Y_j|s) = \frac{p(Y_j)p(s|Y_j)}{p(s)} \tag{5.11}$$

Here $p(Y_j|s)$ is the posterior probability of Y_j given the feature vector s. $p(Y_j)$ is the prior probability of class Y_j which is calculated using the training data. $p(s|Y_j)$ is the conditional probability of the feature vector given the class Y_j. $p(s)$ is the prior probability of the feature vector.

The denominator in the equation 5.11, $p(s)$ has the same value for all the classes j, and hence, can be ignored in naive Bayes classifier while predicting the most likely class of a test sample. So, using the different probabilities which are calculated on the training data, the naive Bayes classifier predicts class label Y for a test sample as below:

$$Y = \underset{j \in \{1,2,...J\}}{\operatorname{argmax}} \, p(Y_j) \prod_{i=1}^{n} p(s_i|Y_j) \tag{5.12}$$

For example, consider a dataset of 5 training samples with three binary features (s_1, s_2, s_3) and a class label Y_j with two possible classes Y_1, Y_2 given in Table 5.2. In the 5 training samples, 3 samples belong to class Y_1 and 2 samples belong to class Y_2. At end of naive Bayes training phase, we calculate the class probabilities $p(Y_1)$ and $p(Y_2)$ as 0.6 and 0.4 respectively, which are proportional to the number of training samples belonging to the given class. In the 3 training samples belonging to class Y_1, let the feature value of s_1 is 1 in two samples and s_1 is 0 in one sample. Then we calculate the conditional probabilities $p((s_1 = 1)|Y_1)$ and $p((s_1 = 0)|Y_1)$ as $\frac{2}{3}$ and $\frac{1}{3}$ respectively. Similarly, in the two samples belonging to class Y_2, the feature value of s_1 is 1 in one sample and s_1 is 0 in the other sample. Then we calculate conditional probabilities $p((s_1 = 1)|Y_2)$ and $p((s_1 = 0)|Y_2)$ as $\frac{1}{2}$ and $\frac{1}{2}$ respectively. The conditional probabilities for all the three features s_1, s_2 and s_3 for their corresponding feature values are given in Table 5.3. Given a test sample s, with naive Bayes we need to predict whether it belongs to the class Y_1 or class Y_2 using the conditional probabilities of its features (s_1, s_2, s_3) and the class probabilities $p(Y_1)$, $p(Y_2)$. Let us consider a test sample $s = (0, 1, 1)$. As per naive Bayes classifier, we calculate the posterior probabilities of the given test sample belonging to class Y_1, Y_2 as $p(Y_1|s = (0, 1, 1))$, $p(Y_2|s = (0, 1, 1))$, where $p(Y_1|s = (0, 1, 1)) = p(Y_1) * p(s_1 = 0|Y_1) * p(s_2 = 1|Y_1) * p(s_3 = 1|Y_1)$ and $p(Y_2|s = (0, 1, 1)) = p(Y_2) * p(s_1 = 0|Y_2) * p(s_2 = 1|Y_2) * p(s_3 = 1|Y_2)$. From the previously computed conditional probabilities and class probabilities using the training data, we can compute $p(Y_1|s = (0, 1, 1)) = 0.6 * \frac{1}{3} * \frac{1}{3} * \frac{1}{3} = 0.022$. Similarly we can compute $p(Y_2|s = (0, 1, 1)) = 0.4 * \frac{1}{2} * \frac{1}{2} * \frac{1}{2} = 0.05$. Since $p(Y_2|s = (0, 1, 1)) > p(Y_1|s = (0, 1, 1))$, naive Bayes predicts that the given sample belongs to class Y_2.

5.4 Proposed approach

This section presents the salient features of our hyper-heuristic approach.

Table 5.2: Sample dataset used for naive Bayes training

	Feature values			Class label
S. No.	s_1	s_2	s_3	Y
1	1	1	0	Y_1
2	0	1	0	Y_2
3	1	0	0	Y_1
4	1	0	1	Y_2
5	0	0	1	Y_1

Table 5.3: Conditional probabilities for different feature values

	Class Y_1		Class Y_2	
Feature	$p(s_i = 0\|Y_1)$	$p(s_i = 1\|Y_1)$	$p(s_i = 0\|Y_2)$	$p(s_i = 1\|Y_2)$
s_1	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{2}$	$\frac{1}{2}$
s_2	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{2}$
s_3	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{2}$

5.4.1 Solution representation and fitness

We represent each solution as a set of p nodes to locate facilities. The objective function given in equation 5.1 is used as the fitness function, which is the weighted sum of the cost of serving customers when there are no facility failures and the expected failure cost which includes the cost of serving customers when there are facility failures.

5.4.2 Generating the initial solution

We have used two methods to generate the initial solutions, viz. a randomized greedy approach in the first iteration of the hyper-heuristic and a random generation method for generating the initial solutions in all the remaining iterations of the hyper-heuristic. In the randomized greedy approach to generate the initial solution, we have made use of the node connectivity of the given input network. In a network, which is an incomplete graph where a given node is not connected to every other node, the facility located at a node can serve only the set of nodes it is connected to. Making use of this nature of the input network, as part of the randomized greedy approach, first we select five nodes with the highest number of adjacent nodes and randomly select one node from these five nodes, and make it part of the solution. The selected node is removed from the list of nodes available for locating facilities. This previous step is repeated till we select a total of p nodes to be part of the initial solution. For the remaining restart iterations of the hyper-heuristic, we generate the initial solutions in a random manner by randomly selecting p unique nodes to be part of the initial solution. In both the randomized greedy approach and

the random generation method of generating the initial solution, there may be some customers which are not connected to any of the p facilities, and hence, remain unserved. For all such unserved customers i, like [142], we have used a fixed cost θ_i as the cost of serving while calculating the fitness value.

5.4.3 Hyper-heuristic framework with naive Bayes classifier

We have implemented a greedy selection based hyper-heuristic approach with multiple starts. In every start, we generate an initial solution, and then apply four low level heuristics in two phases namely training phase and testing phase. The training phase and testing phase are part of the naive Bayes classifier which is deployed in this case to reduce the computation time while improving the solution quality. The four low level heuristics that we have used are described in the subsequent subsection (Section 5.4.4). We will refer to our four low level heuristics as LH_1, LH_2, LH_3, and LH_4 subsequently.

In an iteration of the **training phase**, all the four low level heuristics are applied on the current solution X_{tr} one-by-one thereby creating four new solutions. We have used variable degree of perturbation while applying each of the low level heuristics over many training iterations which varies from Max_{prt}, which is the maximum degree of perturbation, to Min_{prt}, which is the minimum degree of perturbation over the maximum number of training iterations, Max_{tr}. To generate good solutions, we need higher perturbation in the initial iterations and lower perturbation towards the final iterations. In training iteration $iter_{tr}$, the degree of perturbation is calculated as $Deg_{prt} = \left(\frac{Max_{prt}-Min_{prt}}{Max_{tr}}\right)(Max_{tr} - iter_{tr}) + Min_{prt}$ [177]. We select the solution with the least objective value among the four resulting solutions after applying the four heuristics as an input to the next iteration of the training phase. As part of the naive Bayes classification, we treat each of the four low level heuristics as four different classes and in one training phase iteration which ever low level heuristic produces least objective solution, it is considered that the given solution X_{tr} belongs to that particular class. At the end of the training phase, we find the class probabilities of each of the four low level heuristics based on the number of times a low level heuristic generates the solution with least objective value. We calculate the conditional probabilities of feature values of the solution used for for training, X_{tr}, considering a binary value of 1 for all the nodes which are part of the solution and a binary value of 0 for all the nodes which are not part of the solution. Consider an example scenario of a network with 50 nodes and number of facilities to be located $p = 5$. Let a sample solution to this problem with $p = 5$ facilities be $X_{tr} = \{4, 5, 12, 20, 46\}$. So we treat this as a data record

of 50 binary features where the binary value at index i represents whether node i is part of the solution or not. In the given example solution, the feature values at indices $4, 5, 12, 20,$ & 46 are considered as 1s, thereby representing a facility is located at these locations, and in all the other indices feature values are taken as 0s. With this feature vector, we find the conditional probabilities for all the features as explained in Section 5.3. We take the resulting solution at the end of the training phase, X_{tr} as the input to the testing phase, X_{test} and then the testing phase starts.

In each iteration of the **testing phase**, for the given solution X_{test}, we find the posterior probabilities for each of the four low level heuristics using naive Bayes classifier, $p(LH_1|X_{test})$, $p(LH_2|X_{test})$, $p(LH_3|X_{test})$, $p(LH_4|X_{test})$. We will skip the low level heuristic with the least posterior probability, and apply the remaining three heuristics on the solution X_{test} and the degree of perturbation in a test iteration $iter_{test}$ is calculated as $Deg_{prt} = \left(\frac{Max_{prt}-Min_{prt}}{Max_{test}}\right)(Max_{test} - iter_{test}) + Min_{prt}$, where Max_{test} is the maximum number of testing phase iterations. Then we select the solution with the least objective value among the three resulting solutions after applying the three heuristics. Then on the selected solution, we apply the 1-1 exchange local search operation which is explained the subsequent subsection, viz. Section 5.4.5. The resulting solution after local search is considered as an input to the next iteration of the testing phase. If there is no improvement in the best solution objective value for five consecutive test iterations, we stop the testing phase. Once the testing phase ends, we apply 1-1 exchange on the best solution found so far and then the hyper-heuristic makes a fresh start. The pseudo-code of our approach is presented in Algorithm 9.

There are several acceptance criteria discussed in the hyper-heuristic literature [74] for selecting the newly generated solution as current solution in the next iteration. In our proposed approach, we have experimented with two acceptance criteria namely AA (all acceptance), OI (only improvement). The AA criterion always selects the newly generated solution, whereas the OI criterion selects the newly generated solution only if it is better than the current solution. In our experiments, we have found that the AA criterion provided better results in comparison to the OI criterion. This can be attributed to the better exploration of the search space as a result of following the AA criterion due to the selection of a new solution in every iteration. Hence, we have used the AA criterion in both the training and testing phases of our proposed hyper-heuristic approach.

Algorithm 9: Pseudo-code of hyper-heuristic for RpMP

Input: List of parameters for greedy hyper-heuristic and an instance of RpMP
Output: Overall best solution found
$X_{best} \leftarrow \emptyset$;
while *termination condition remains unsatisfied* **do**
  $X_{init} \leftarrow$ Generate initial solution;
  $X_{tr} \leftarrow X_{init}$;
  while *($iter_{tr} \leq Max_{tr}$)* **do**
    $Deg_{prt} \leftarrow \left(\frac{Max_{prt}-Min_{prt}}{Max_{tr}}\right)(Max_{tr} - iter_{tr}) + Min_{prt}$;
    $X_1 \leftarrow LH_1(X_{tr}, Deg_{prt})$;
    $X_2 \leftarrow LH_2(X_{tr}, Deg_{prt})$;
    $X_3 \leftarrow LH_3(X_{tr}, Deg_{prt})$;
    $X_4 \leftarrow LH_4(X_{tr}, Deg_{prt})$;
    $X_{tr} \leftarrow MIN(X_1, X_2, X_3, X_4)$;
    if *(X_{tr} is better than X_{best})* **then**
      $X_{best} \leftarrow X_{tr}$;
  $X_{test} \leftarrow X_{tr}$;
  while *($iter_{test} \leq Max_{test}$)* **do**
    $skip\_hr \leftarrow Naive\_Bayes(X_{test})$;
    $Deg_{prt} \leftarrow \left(\frac{Max_{prt}-Min_{prt}}{Max_{test}}\right)(Max_{test} - iter_{test}) + Min_{prt}$;
    for *($i := 1$ to 4)* **do**
      if *($skip\_hr! = i$)* **then**
        $X_i \leftarrow LH_i(X_{test}, Deg_{prt})$;
    $Fitness(X_{skip\_hr}) \leftarrow RAND\_MAX$;
    $X_{test} \leftarrow MIN(X_1, X_2, X_3, X_4)$;
    $X_{test} \leftarrow Local\_search(X_{test})$;
    if *(X_{test} is better than X_{best})* **then**
      $X_{best} \leftarrow X_{test}$;
  $X_{best} \leftarrow Local\_search(X_{best})$;
return X_{best};

5.4.4 Low level heuristics

We have used the following four low level heuristics in our hyper-heuristic approach:

- LH_1**: Random removal and greedy addition**
  In the first low level heuristic, from the given solution sol containing p facilities, a given number $NR(= p * Deg_{prt})$ facilities are removed randomly. Now, to again get a feasible solution with p facilities, we need to add NR nodes back to the solution. These NR nodes are added in an iterative manner where during each iteration we select a node with the maximum adjacency count from all the nodes which are not currently part of the

solution, and add it to the solution.

- LH_2**: Random addition and greedy removal**
  As part of the second low level heuristic, from the list of nodes which are not part of the given solution sol containing p facilities, a fixed number $NA(= p * Deg_{prt})$ of nodes are randomly selected and added to the current solution. Now, the solution contains $p + NA$ facilities. To remove the extra NA facilities from sol, we eliminate one facility at a time from the solution sol in an iterative manner till there are only p facilities remaining. In each iteration, the facility with minimum adjacency count is chosen for removal.

- LH_3**: Greedy removal and greedy addition**
  In the third low level heuristic, from the given solution with p facilities, we iteratively select a facility with the least adjacency count and remove it from the solution. In this fashion, a total of $NR(= p * Deg_{prt})$ facilities are removed from the given solution. Then, from all the locations which are not part of the solution, we select a node with the maximum adjacency count and add it to the solution. In this manner, nodes are added iteratively till the total number of facilities in the solution reaches p.

- LH_4**: Greedy addition and greedy removal**
  In the fourth low level heuristic, to the given solution having p facilities, we add $NA(= p * Deg_{prt})$ new facilities in a greedy manner. This is an iterative procedure where in each iteration, out of all the locations which are not part of the solution, we select a location with the maximum adjacency count and add it to the solution. At the end of this iterative process, the solution contains $p + NA$ facilities. Then, we select a facility with the least adjacency count and remove it from the solution. This process is repeated and is stopped when there are only p facilities remaining in the solution.

5.4.5 Local search

As part of the local search, we have implemented a 1-1 exchange where for each facility in the given solution, we try to find a replacement node from the set of nodes which are not part of the solution. For this, a facility is considered for replacement with all $N - p$ nodes which are not part of the solution and is replaced with the node that provides the solution with the least objective value [178]. A facility which is replaced from the solution with a new node, can be reintroduced into the solution at a different index. For a facility under consideration if there

is no replacement node available whose addition to the solution in place of the given facility results in a reduction in the objective value, then that facility continues to be part of the solution and exchange of that facility does not happen. And the next facility in the solution is considered for replacement. This iterative process is repeated for all p facilities in the solution. Within the testing phase, we apply the 1-1 exchange for the initial two test iterations and also when the current solution objective value is less than the current best solution objective value. But for the cases when the current solution objective value is greater than that of the best solution, we apply the 1-1 exchange only if the difference is within 20% of the best solution's objective value. We have arrived at this 20% after a large number of experiments.

5.5 Computational results

We have implemented the proposed hyper-heuristic approach in C. In all our experiments, we have taken the maximum number of training iterations, $Max_{tr} = 15$ and the maximum number of test iterations, $Max_{test} = 10$. We have taken the maximum perturbation $Max_{prt} = 0.4$, and minimum perturbation $Min_{prt} = 0.1$. We have chosen these parameter values empirically after a large number of experiments. The cost of not serving a customer i by any facility, θ_i is taken as 300.0 just as used in [142]. We have also conducted our experiments with two values of α, 0.2 and 0.8. We have considered the same failure probability $q = 0.05$ and the demand of each customer $h_i \forall i \in A$ is taken as 1 just like in [142].

We have compared the results of our approach with the state-of-the-art methods for RpMP proposed in [142]. Just like in [142], we have also executed our approach on the same two datasets namely Discrete Location Problems Library (DLPL)[1] and the OR library[2] which Beasley [179] introduced. We have considered the termination condition of our approach as the maximum execution time in seconds which is a combination of the number of nodes n and the number of facilities to be located p and is calculated as $2 * \ln \binom{n}{p}$, just like in [142]. We have executed our hyper-heuristic approach on a Linux based 3.10 GHz Core-i5-8600 system with 8 GB RAM. We have reported our results with the naive Bayes classifier and without the naive Bayes classifier also to show the benefit of former over latter.

As part of the experimentation in [142], they have solved all the instances using CPLEX executing each instance for a maximum cpu time of 2 hours. In allotted time of two hours,

[1] http://math.nsc.ru/AP/benchmarks/UFLP/

[2] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/esteininfo.html

for some instances CPLEX found the optimal solution, for some CPLEX found a feasible solution and for some CPLEX even fails to find even any feasible solution. To evaluate the performance of genetic algorithm and scatter search approaches proposed in [142], they have calculated the percentage deviation of the objective values of their approaches from the solution obtained by CPLEX at the end of two hours if CPLEX was able to find any feasible solution. Otherwise, the deviation is computed from the best solution among all the 8 genetic algorithm and scatter search based approaches. Out of the total 8 variations of the methods proposed in [142], they have reported GA2 and SS2 to be the best performing methods. We have obtained the instance-by-instance results from the first author of [142] through personal communication, and reported on each instance, the percentage deviation of the objective value of our hyper-heuristic approach with respect to the objective value of the solution used as reference in [142] for that instances. We have compared our results with the results of only GA2 and SS2 which are shown to be the best performing methods. Table 5.4 gives the comparison of results on DLPL instances, Table 5.5 gives the comparison of results on OR library instances where CPLEX has reached a feasible solution, and Table 5.6 gives the comparison of results on OR library instances where CPLEX has not reached a feasible solution. In all of the three tables, viz. Table 5.4, Table 5.5, and Table 5.6, first column N represents the number of nodes in the network, second column p gives the number of facilities to be located. The values in each column under the heading %Deviation report the the percentage deviation of the corresponding approach's objective value from the objective value of the optimal/best solutions obtained by CPLEX on the same datasets. The 3rd, 4th, 5th and 6th columns give the percentage deviation of the objective values obtained by GA2, SS2, hyper-heuristic without naive Bayes and hyper-heuristic with naive Bayes respectively. The values in columns under the heading CPU Time, i.e., values in the 7th, 8th, 9th and 10th columns provide the time taken in seconds to arrive at the best solution by the GA2, SS2, hyper-heuristic without naive Bayes and hyper-heuristic with naive Bayes respectively. In the Table 5.4, the row which starts with TOTAL 50 gives the average deviation and average CPU time for the instance with 50 nodes for different values of p. Similarly the row which starts with TOTAL 100 gives the average deviation and average CPU time for the instance with 100 nodes for different values of p. The last row in the Table 5.4, which starts with OVERALL gives the average deviation and average CPU time over all the instances calculated as the average of the values from rows starting with TOTAL 50, TOTAL 100. We have followed the same method of reporting in the other tables, viz. Table 5.5, and Table 5.6 as well.

As shown in Table 5.4, on the DLPL instances our hyper-heuristic with naive Bayes achieved an average deviation of -1.21% as against the average deviation of 0.32% of GA2 and 0.02% of SS2 from the CPLEX solution. Even our hyper-heuristic approach without naive Bayes achieved an average deviation of -1.09% which is better than that of the GA2 and SS2. For instances with 50 nodes our hyper-heuristic without naive Bayes performed better than the hyper-heuristic with naive Bayes. Overall, on combining the results of instances with 50 nodes and 100 nodes of DLPL instances, our hyper-heuristic with naive Bayes outperforms our hyper-heuristic without naive Bayes.

On the OR library instances where CPLEX has reached feasible solution in Table 5.5, our hyper-heuristic with naive Bayes achieved an average deviation of -7.3% as against the average deviation of -5.41% of GA2 and -7.06% of SS2 from the CPLEX solution. On the same set of instances, the deviation achieved by our hyper-heuristic without naive Bayes (-7.05%) is slightly worse as compared to that of the SS2(-7.06%), but still better than that of the GA2 (-5.41%). Overall, our hyper-heuristic with naive Bayes performed better than GA2, SS2 and hyper-heuristic without naive Bayes on the instances where CPLEX has reached a feasible solution. On the OR library instances where CPLEX has not reached a feasible solution in Table 5.6, our hyper-heuristic approach with naive Bayes and without naive Bayes have performed better than GA2, SS2.

The effectiveness of using naive Bayes can be observed from the improved overall solution quality and reduced computation time as compared to when there is no naive Bayes classifier as can be observed from the overall average deviations and CPU times reported in Tables 5.4, 5.5 and 5.6. The system configuration used for executing our approach is different from the one used in [142] and hence we can't directly compare the execution times. However, from the overall execution times it is clear that our proposed hyper-heuristic with naive Bayes and without naive Bayes are slower in comparison to the existing approaches of GA2 and SS2.

Table 5.4: Comparison of results given by the hyper-heuristic with GA and SS (DLP Library)

		%Deviation				CPU Time			
N	p	GA2	SS2	HH(without NB)	HH(with NB)	GA2	SS2	HH(without NB)	HH(with NB)
50	5	0.77	1.01	**0**	**0**	0.33	0.29	0.39	0.41
50	10	2.06	0.48	**0**	0.21	8.21	0.56	3.03	1.96
50	15	0.36	0.36	**0.06**	**0.06**	2.27	1.1	5.56	3.76
TOTAL 50		1.07	0.62	**0.02**	0.09	3.61	0.65	2.99	2.04
100	5	0.37	0.46	-0.53	**-1.02**	1.82	1.54	1.67	1.05
100	10	-0.43	-1.22	**-3.59**	-3.55	12.46	6.6	16.00	12.82
100	15	-1.23	-0.95	-2.48	**-2.93**	20.47	18.4	29.42	33.63
TOTAL 100		-0.43	-0.57	-2.2	**-2.5**	11.59	8.85	15.70	15.83
OVERALL		0.32	0.02	-1.09	**-1.21**	7.60	4.75	9.35	8.94

Table 5.5: Comparison of results given by the hyper-heuristic with GA and SS (OR Library: Instances where CPLEX has reached a feasible solution)

		%Deviation				CPU Time			
N	p	GA2	SS2	HH(without NB)	HH(with NB)	GA2	SS2	HH(without NB)	HH(with NB)
100	5	**0**	**0**	**0**	**0**	0.08	0.13	0.02	0.01
100	10	**0**	0.03	**0**	**0**	2.3	0.52	1.49	0.92
100	20	**0**	0.43	1.25	**0**	45.42	13.62	44.60	23.15
100	20	0.99	0.91	1.31	**0.81**	45.62	4.36	30.59	30.85
100	33	0.14	0.46	**0.02**	**0.02**	33.09	10.18	77.64	72.05
TOTAL 100		0.22	0.37	0.52	**0.17**	25.30	5.76	30.87	25.4
200	5	**0**	**0**	**0**	**0**	0.42	0.50	0.05	0.03
200	10	**0**	0.04	**0**	**0**	18.93	6.59	5.42	13.41
200	20	-9.74	-9.88	**-9.97**	**-9.97**	103.36	47.32	24.84	14.57
200	40	6.22	1.78	1.56	**1.33**	174.7	92.76	81.47	95.00
200	67	0.9	**0.29**	0.38	0.91	206.21	90.94	110.68	113.86
TOTAL 200		-0.52	-1.55	**-1.61**	-1.55	100.72	47.62	44.49	47.37
300	5	**0**	**0**	**0**	**0**	0.76	1.11	112.09	0.06
300	10	**0**	**0**	**0**	**0**	28.69	4.98	52.52	3.33
300	30	-83.74	**-84.75**	-84.55	-84.54	146.6	135.67	112.09	66.11
300	60	-21.4	-31.67	-31.17	**-32.23**	238.9	227.32	265.88	257.56
300	100	14.08	0.38	1.92	**0.21**	334.84	339.61	341.88	370.53
TOTAL 300		-18.21	-23.21	-22.76	**-23.31**	149.96	141.73	176.89	139.52
400	5	**0**	**0**	**0**	**0**	3.45	2.05	0.19	0.11
400	10	**-0.27**	-0.25	-0.26	-0.26	37.1	15.10	11.33	39.78
400	133	-9.98	-11.89	-14.43	**-14.92**	349.4	559.20	725.89	578.74
TOTAL 400		-3.42	-4.05	-4.90	**-5.06**	129.98	192.12	245.80	206.21
500	5	**0**	**0**	**0**	**0**	3.66	3.2	0.29	0.17
OVERALL		-5.41	-7.06	-7.05	**-7.3**	93.34	81.85	105.21	88.43

Table 5.6: Comparison of results given by the hyper-heuristic with GA and SS (OR Library: Instances where CPLEX has not reached a feasible solution)

		%Deviation				CPU Time			
N	p	GA2	SS2	HH(without NB)	HH(with NB)	GA2	SS2	HH(without NB)	HH(with NB)
400	40	23.57	2.59	5.80	**0.95**	202.77	211.21	189.09	146.20
400	80	28.83	1.18	**0.48**	1.08	281.02	375.76	426.81	261.49
TOTAL 400		26.20	1.88	3.14	**1.01**	241.9	293.48	307.95	203.85
500	10	0.13	**0**	0.24	0.24	69.35	43.37	75.24	12.38
500	50	10.07	1.52	**-0.09**	0.04	260.82	311.99	269.57	322.40
500	100	27.25	0.45	**-3.33**	-1.87	286.73	533.27	651.79	621.57
500	167	6.99	**1.82**	4.51	4.04	561.21	654.99	1037.61	859.36
TOTAL 500		11.11	0.95	**0.33**	0.61	294.53	385.9	508.55	453.92
600	5	**0**	**0**	**0**	**0**	3.48	4.44	1.22	0.48
600	10	0.46	0	**-0.01**	**-0.01**	58.20	19.87	24.03	1.52
600	60	14.37	3.13	-0.88	**-2.45**	264.49	407.79	448.91	311.17
600	120	25.53	1.94	2.49	**-1.71**	525.04	677.22	754.92	1174.38
600	200	24.91	1.02	**-0.24**	-0.14	1021.13	977.4	2451.07	2078.27
TOTAL 600		13.05	1.22	0.27	**-0.86**	374.47	417.34	736.03	713.16
700	5	0	**0**	0.01	0.01	5.48	6.09	0.58	0.33
700	10	0.2	**0**	0.26	0.71	72.63	23.66	13.83	87.43
700	70	11.38	4.76	0.95	**-0.02**	366.20	461.3	603.92	535.99
700	140	30.47	0.82	**0.78**	**0.78**	744.83	921.46	1526.23	1227.65
TOTAL 700		10.51	1.39	0.50	**0.37**	297.28	353.13	536.14	462.85
800	5	**0**	**0**	**0**	**0**	9.13	7.86	4.31	1.28
800	10	0.34	0.01	**0**	**0**	72.55	38.89	9.09	5.34
800	80	23.69	0.9	3.62	**-1.99**	370.13	590.44	538.19	787.29
TOTAL 800		8.01	0.3	1.21	**-0.66**	150.6	212.39	183.86	264.64
900	5	0	0	**-0.01**	**-0.01**	14.54	11.13	0.94	0.55
900	10	0.45	**0.13**	0.24	0.22	58.91	70.84	23.22	91.4
900	90	6.53	1.21	**-1.04**	0.64	453.55	634.3	910.95	680.72
TOTAL 900		2.33	0.45	**-0.27**	0.28	175.66	238.75	311.70	257.56
OVERALL		11.2	1.02	0.66	**0.02**	271.53	332.53	474.36	438.44

5.6 Conclusions

In this chapter, we have proposed a hyper-heuristic based approach with naive Bayes classifier for the reliability p-median problem (RpMP). We have compared the results of our approach with the state-of-art approaches available in the literature [142]. The effectiveness of our approach can be observed in terms of the solution quality. However, our approach is slower compared to the existing approaches. Ours is the maiden hyper-heuristic approach proposed for solving the RpMP.

Chapter 6

Reliable p-median problem with at-facility service

6.1 Introduction

In this chapter we discuss another variant of the p-median problem which considers the facility failures, namely reliable p-median problem with at-facility service (RpMF). Just like the reliability p-median problem (RpMP) which is dicussed in the previous chapter, RpMF is also concerned with locating p facilities in such a manner that minimizes the total cost while taking into consideration the cost of facility failures. However, RpMF differs from RpMP in that for each customer considering the current facility failure, the next closest facility from the failed facility is considered as the backup facility in RpMF. On the other hand, in RpMP when a facility assigned to the customer fails, the next closest facility from the customer's starting location is considered as the backup facility. Consider the scenarios where the service is given at the customer's location or even though the service is at the facility and the customer knows about the facility failure then in both these cases the customer can be serviced from the next closest facility from the customer's starting location. RpMP comes under this category of facility location models. There can be scenarios where the service is at the facility and the customer doesn't know about the facility status until he/she visits it. In such cases, the customer chooses a new facility with respect to the location of the currently inoperable facility, rather than from the customer's starting position. RpMF comes under this latter category of facility location models. Berman et al. [180] discussed this model of locating facilities where facilities may be inoperable and the customers don't have prior knowledge about the facility status.

In real world, such scenarios arise in several day-to-day situations. Some of examples of this model finding its applications in daily life are as follows: bank customers withdrawing cash by visiting their nearest ATM point on regular basis which may not be servicing customers at a given time due to maintenance of the machine, people visiting petrol filling stations that have long waiting queues or shortage of petrol, patients visiting hospitals in emergency condition and are forced to seek treatment elsewhere due to long waiting times and the other example being customers visiting retail stores to purchase a necessary item which may not be available in the store. In all the just mentioned examples, we can assume that the customers do not have complete information about the facility status, and hence, visit the other facilities in trial and error method usually based on shortest distance from the current location till successfully receiving the service at a functioning facility or in some cases even stop visiting facilities after some unsuccessful attempts. When identifying the next facility to visit after a facility failure, a customer may either chose the next nearest facility as used in [180] or apply an optimized search scheme based on the less expected distance to be traveled before receiving the desired service. As an application of optimized search scheme, consider the case of military operations in a war situation where due to lack of proper communication military units are wanting to reach facilities that may be under attack and unable to provide services. In such scenarios, there should be well thought through contingency plans with an aim to reduce the expected travel distance of the troops thereby improving the chance of survival of the army personnel. Consider the other application, the case of natural disasters like earthquake or floods in an area. In such cases, it is difficult to get the live information due to damage of communication channels or failure of facilities. And also the emergency facilities located in such places may get damaged. Hence, there should be contingency plans in place to efficiently carry out the rescue operations. In [181], Albareda-Sambola et al. worked on the reliable p-median problem with at facility service (RpMF) with the assumption that the customers do not have complete information about the facility failures and that the customers follow an optimized search scheme to find a facility to receive service. Albareda-Sambola et al. came up with two different mathematical programming formulations for RpMF which are termed F1, F2, and also a network flow model based matheuristic for the RpMF [181]. The first mathematical formulation, F1, is the general case where site-dependent failure probabilities of the facilities are considered. The second mathematical formulation, F2, is termed as 'binary formulation for the homogeneous case' and assumes equal or homogeneous probabilities of failure for all the failable facilities. Hence, F2 is not valid for the case of non-homogeneous probabilities of failure. The other formulation

considered in [181] is termed as 'flow approximation to the RpMF' or 'formulation FP'. This formulation, FP, assumes that a customer can revisit any of the opened p facilities, and hence, doesn't require to maintain the path of the customer. Between the formulations F1 and F2, F2 has higher space requirements and generates solutions in reasonable times whereas F1 can be used to solve larger instances as it has less memory requirements but takes long execution times. On the other hand, FP generates heuristic solutions to the RpMF and has much less space requirements than F1 and can be solved in less execution times than F2.

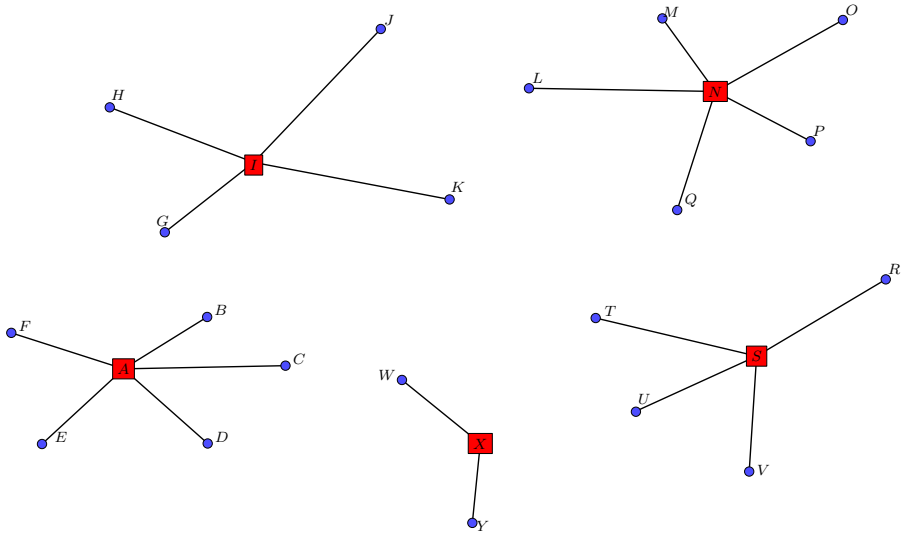
Consider Figure 6.1 that depicts a sample network with the number of nodes $n = 25$ and number of facilities $p = 5$ that illustrates the RpMF model considering a facility failure. We have shown the facilities in rectangle shape and marked in red color for ease of identification. The sub-figure 6.1a shows assignment of nodes to their nearest facilities. Nodes B, C, D, E, F are assigned to their nearest facility A. Nodes G, H, J, K are assigned to facility I. Similarly nodes L, M, O, P, Q are assigned to facility N and nodes R, T, U, V are assigned to facility S. Finally, nodes W, Y are assigned to facility X. Suppose, if the facility X fails then all the nodes that are assigned to it, namely W, Y including the facility X are now re-assigned to the next closest functioning facility from X which is S as shown in sub-figure 6.1b. The re-assignment of nodes W, X, Y to the new facility is depicted in dotted lines in sub-figure 6.1b.

In the example of Figure 6.1 between the two functioning facilities A and S, the starting location of customer W is closer to facility A than facility S. If it were RpMP, then W would have been re-assigned to A instead of S. So, this illustration explains that even though RpMF and RpMP are related, they are two different facility location models. The former is appropriate when service is provided at the facility and a customer can not know the status (failed/operational) of the facility till he/she visits it. On the other hand, if customer can know the status of the facility from his/her location or when service is provided at customer's location then RpMP is appropriate.

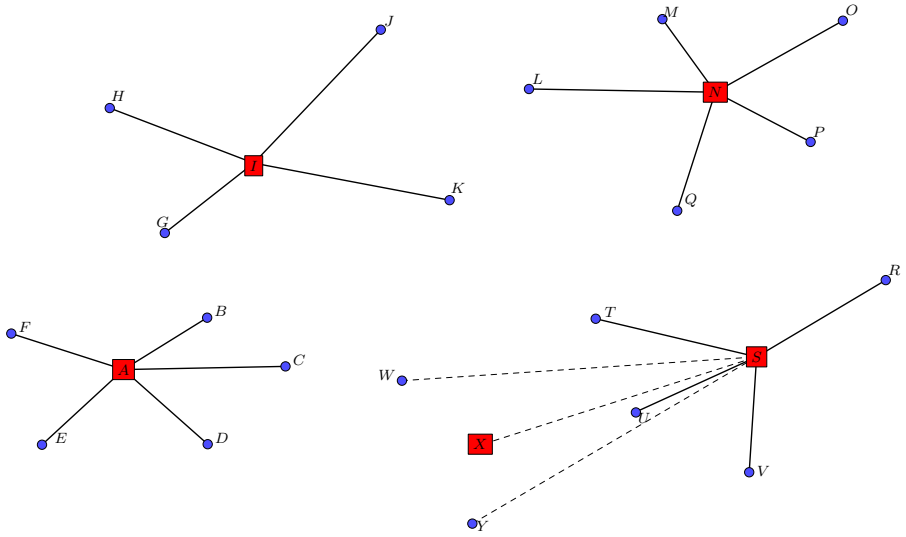
The three mathematical programming based approaches, viz. F1, F2 and FP presented in [181] are the only approaches available in the literature for RpMF. In spite of its practical applicability, RpMF remains an understudied problem. No problem-specific heuristic and metaheuristic approaches exists in the literature for RpMF. This has motivated us to work on RpMF and develop the approaches presented in this chapter. We have proposed two multi-start hyper-heuristic based approaches to solve the RpMF and evaluated the performance of our approaches on the same datasets as used in [181]. We have compared our results with the optimal or best known solution values reported by [181]. We have shown the effectiveness of

our proposed approaches as our approaches obtain high quality solution in much less execution times for all instances.

The rest of this chapter is organized in the following manner: Section 6.2 gives the formal mathematical definition of RpMF, Section 6.3 describes the proposed approaches in detail, Section 6.4 presents the computational results and analyses them. Finally, Section 6.5 concludes the chapter by presenting the summary of contributions made.



(a) Node assignments when all the five facilities are functioning



(b) Node assignments when one facility fails

Figure 6.1: Illustration of reliable p-median problem with at facility service having $p = 5$ facilities

6.2 Formal problem definition

We have given the formal definition of RpMF using notational conventions similar to the one used in [181]. We have provided the summary of key notations in Table 6.1.

Table 6.1: Summary of key notations

Notation	Meaning
A	Set of clients or demand points which is also the set of potential facility locations
F	Subset of potential facility locations which may fail
NF	Subset of potential facility locations which will never fail
h_i	Demand associated with each client or demand point $i \in A$
d_{ij}	Shortest distance between two locations i, j
q_j	Probability of failure of the facility $j \in F$
p	Total number of opened facilities

The other required variables s_i, x_{ijk} and y_{ijk} are defined as below,

$$s_i = \begin{cases} 1 & \text{if the location } i \text{ is selected to locate a facility, } \ i \in A \\ 0 & \text{Otherwise} \end{cases}$$

$$x_{ijk} = \begin{cases} 1 & \text{if customer } i \text{ passes from location } j \text{ to location } k, \ i, j, k \in A \\ 0 & \text{Otherwise} \end{cases}$$

$y_{ijk} \in [0, 1]$: probability of the edge (j, k) to be in the path of customer i

Using the notations given in Table 6.1 and the variables s_i, x_{ijk} and y_{ijk}, RpMF can be mathematically formulated [181] as follows:

$$min \ \sum_{i \in A} h_i \sum_{j \in A} \sum_{k \in A} d_{jk} y_{ijk} \tag{6.1}$$

subject to

$$\sum_{j \in A} s_j = p \tag{6.2}$$

$$s_1 = 1 \tag{6.3}$$

$$\sum_{j \in A} x_{ijk} \leq s_k \ \ i, k \in A \tag{6.4}$$

$$\sum_{k \in A} x_{ijk} \leq 1 \;\; i, j \in A \tag{6.5}$$

$$\sum_{k \in A} y_{ijk} = q_j \sum_{k' \in A} y_{ik'j} \;\; i \in A, j \in F, i \neq j \tag{6.6}$$

$$\sum_{j \in A, j \neq i} y_{iik} = (1 - s_i) \;\; i \in NF \tag{6.7}$$

$$\sum_{j \in A, j \neq i} y_{iik} = 1 - (1 - q_i)s_i \;\; i \in F \tag{6.8}$$

$$y_{ijk} \leq x_{ijk} \;\; i, j, k \in A \tag{6.9}$$

$$s_j, x_{ijk} \in \{0, 1\} \;\; i, j, k \in A \tag{6.10}$$

$$y_{ijk} \in [0, 1] \;\; i, j, k \in A \tag{6.11}$$

Equation 6.1 represents the objective function of the problem. Constraint in equation 6.2 makes sure that exactly p facilities are located. Equation 6.3 defines a dummy facility being located at the 1st node which is always non-failable and is used to serve customers which are not served by any of the original p facilities. By assuming the existence of a dummy facility, the problem actually becomes the one concerning locating $p + 1$ facilities in a $n + 1$-node network. But, following the traditional representation as mentioned in [181], we don't include the dummy facility in both the total number of nodes in the network and the number of facilities to be located. Constraint in equation 6.4 requires that a customer i can reach a facility k atmost once. Similarly, the constraint in equation 6.5 requires that a customer i can leave a facility j atmost once. Constraint 6.6 enforces that the probability of a customer i leaving a facility j is the product of failure probability of facility j, q_j times the probability of customer i reaching the facility j. Constraints 6.7, 6.8 give the probabilities corresponding to customer i leaving its home facility when it is non-failable and failabale respectively. Constraint 6.9 makes sure that the values of the x, y variables are consistent. Constraint 6.10 enforces the binary nature of the s, x variables, whereas constraint 6.11 provides the range of probability values for the variable y.

6.3 Proposed approach

This section describes our proposed two hyper-heuristic approaches for the RpMF. In the subsections that follow, we will describe various features of our approaches.

6.3.1 Solution representation and fitness

A solution to the RpMF is represented as a set of p nodes where facilities can be located. We have directly used the objective function given in equation 6.1 as the fitness function.

6.3.2 Initial solution generation

To generate an initial solution, we have employed the semi greedy method introduced in [182] for p-center problem which makes use of critical distances. Critical distance is defined as the maximum distance among all the distances from nodes to their respective nearest facilities. The node whose distance to its nearest facility gives the critical distance is called the critical node. Just like in [182], we have done pre-processing on the given input data. The pre-processing is a two step procedure which is explained below:

- As part of the first step of the pre-processing, we find the shortest distance from each node to all other nodes using the Floyd–Warshall algorithm, represented by the matrix M. We sort elements in each row of the distance matrix M, along with their respective indices in non-decreasing order and generate two new matrices M_{sort} and $M_{circles}$. The row-wise sorted elements of M are stored in the matrix M_{sort}, while row i in the matrix $M_{circles}$ contains the information on the $1^{st}, 2^{nd}, \ldots, n^{th}$ nearest node to node i. For the sake of explanation, consider the following example with a total of $n = 5$ nodes.

$$M = \begin{bmatrix} 0 & 4 & 7 & 9 & 2 \\ 4 & 0 & 3 & 8 & 1 \\ 7 & 3 & 0 & 6 & 5 \\ 9 & 8 & 6 & 0 & 7 \\ 2 & 1 & 5 & 7 & 0 \end{bmatrix}$$

  The matrices M_{sort} and $M_{circles}$ corresponding to matrix M above are

$$M_{sort} = \begin{bmatrix} 0 & 2 & 4 & 7 & 9 \\ 0 & 1 & 3 & 4 & 8 \\ 0 & 3 & 5 & 6 & 7 \\ 0 & 6 & 7 & 8 & 9 \\ 0 & 1 & 2 & 5 & 7 \end{bmatrix}$$

$$M_{circles} = \begin{bmatrix} 1 & 5 & 2 & 3 & 4 \\ 2 & 5 & 3 & 1 & 4 \\ 3 & 2 & 5 & 4 & 1 \\ 4 & 3 & 5 & 2 & 1 \\ 5 & 2 & 1 & 3 & 4 \end{bmatrix}$$

- As part of the second step of the pre-processing, we calculate a new matrix M_{radius}. We set each element in the principal diagonal of M_{radius} to 1. Every other element (i, j) of M_{radius} where $i \neq j$ gives the number of nodes which are at a lesser distance from the node i than the distance between the nodes i and j. The M_{radius} matrix for the considered example is given below:

$$M_{radius} = \begin{bmatrix} 1 & 2 & 3 & 4 & 1 \\ 3 & 1 & 2 & 4 & 1 \\ 4 & 1 & 1 & 3 & 2 \\ 4 & 3 & 1 & 1 & 2 \\ 2 & 1 & 3 & 4 & 1 \end{bmatrix}$$

In order to generate an initial solution, the first facility is selected randomly and added to the initial solution. The remaining $p - 1$ facilities are added to the initial solution iteratively one-by-one. To select the next facility, we find the critical distance and also identify the critical node. Considering the critical distance as radius and the critical node as the center, a circle is constructed. From all the nodes within this circle, a node is randomly selected and is added to the initial solution as the next facility. To achieve the just mentioned functionality, we make use of the matrices constructed in the pre-processing step. For a given pair of critical node x and its nearest facility y, $l = M_{radius}(x, y)$ gives the number of nodes which are within the critical distance from the critical node. These l nodes are the first l elements in the i^{th} row of the $M_{circles}$ matrix. We randomly select a number r between $1, l$ and make the node $M_{circles}(i, r)$ as the next facility. So, we can summarize that given the critical node x and its nearest facility y, the new facility y' is identified as: $y' = M_{circles}(x, rand(1, M_{radius}(x, y)))$. After adding a

new facility to the solution, we find the latest critical distance and the corresponding critical node and proceed to identify the next facility. This procedure is repeated till we get p facilities in the initial solution.

6.3.3 Hyper-heuristic framework

We have proposed two multi-start hyper-heuristic approaches for the reliability p-median problem with at-facility service (RpMF). Except for the selection mechanism both the proposed approaches follow the same framework. In each restart of the hyper-heuristic, we generate an initial solution which acts as the starting current solution, Sol_{cur} and then an iterative process ensues for N_{itrs} iterations. In each of the N_{itrs} iterations, we generate a new solution by making use of the one or more of the five low-level heuristics which are applied as per the selection mechanism given in subsection 6.3.6. We have described each of the five low-level heuristics which we have implemented as part of the proposed approach in subsection 6.3.4. In each of the low-level heuristics, we have used the same degree of perturbation, D_{prt}. The newly generated solution Sol which is obtained using low-level heuristics either replaces the current solution Sol_{cur} or is discarded as per the acceptance criteria mentioned in subsection 6.3.7. We compare the fitness of the Sol_{cur} with that of the best solution found so far. If the fitness of Sol_{cur} is less than that of the best solution Sol_{best}, then Sol_{cur} replaces the Sol_{best}. Then, Sol_{cur} is given as input to the next iteration. After applying the low-level heuristics for a total of N_{itrs} iterations, at the end of the restart we try to further minimize the fitness of Sol_{best} by performing local search operation which is explained in subsection 6.3.5. Then one restart iteration of the hyper-heuristic is complete and the next restart of the hyper-heuristic starts. This procedure stops after N_{rst} restarts. At the end of the algorithm, we return the the best solution as the output of the proposed approach. Algorithm 10 presents the common framework of both our proposed hyper-heuristics where the function $Selection\_Mechanism()$ distinguishes the two proposed approaches. The function $Selection\_Mechanism()$ takes current solution Sol_{cur} and the set of low-level heuristics S_{LH} as inputs and returns a solution Sol.

6.3.4 Low level heuristics

We have used the following five low-level heuristics in our hyper-heuristic approaches:

- LH_1**: Remove randomly and add greedily**
  In this low level heuristic, we randomly remove one facility at a time from the given

Algorithm 10: Hyper-heuristic approach for the RpMF

Input: List of parameters for the hyper-heuristic and an instance of RpMF
Output: Overall best solution found
$Sol_{best} \leftarrow \emptyset$;
while *(*$itr_{rst} \leq N_{rst}$*)* **do**
  $Sol_{init} \leftarrow$ Generate initial solution;
  $Sol_{cur} \leftarrow Sol_{init}$;
  while *(*$iter \leq N_{itrs}$*)* **do**
    $Sol \leftarrow Selection\_Mechanism(Sol_{cur}, S_{LH})$;
    if *(*Sol *is better than* Sol_{cur}*)* **then**
      $Sol_{cur} \leftarrow Sol$;
    if *(*Sol_{cur} *is better than* Sol_{best}*)* **then**
      $Sol_{best} \leftarrow Sol_{cur}$;
  $Sol_{best} \leftarrow Local\_search(Sol_{best})$;
return Sol_{best};

solution Sol, till a total of x facilities are removed. The value of x is calculated as a fraction of the total number of facilities p, $x = p * D_{prt}$. After removing x facilities randomly, we greedily add the same x number of facilities to the solution Sol. Out of all the non-facility nodes, we select a node whose addition to Sol causes the least increase in the fitness value and add it to the solution Sol. This aforementioned step is repeated iteratively till x facilities are added to the solution Sol and there are p facilities at the end of this heuristic.

- LH_2**: Add randomly and remove greedily**

  In the second low level heuristic, from the non-facility nodes, we randomly add x new facilities to the given solution Sol. x is a fraction of the total number of facilities p, $x = p * D_{prt}$. After adding x new facilities, the total number of facilities in Sol is now equal to $p + x$. After this, we greedily remove one facility at a time from Sol till there are only p facilities remaining. Out of the $p + x$ facilities in Sol, we select a facility whose removal causes maximum reduction in the objective value and remove that facility from Sol. We continue to remove facilities just like in the aforementioned greedy method and stop once a total of x facilities are removed.

- LH_3**: Remove greedily and add greedily**

  As part of the third low level heuristic, from the given solution Sol, we greedily remove one facility at a time till a total of x facilities are removed. Just like in the other low level

heuristics, x is calculated as $x = p * D_{prt}$. From the p facilities in Sol, we select a facility whose removal causes maximum reduction in the objective value and remove that facility from Sol. Once the x facilities are removed, we greedily add one facility at a time till there are p facilities in the resulting solution. For this, from the list of non-facility nodes we select a node whose addition to Sol causes the least increase in the fitness value and add it to the solution Sol. We continue to add further facilities to Sol in the same manner till there are total of p facilities in Sol.

- LH_4**: Add greedily and remove greedily**
  As part of the fourth low level heuristic, to the given solution Sol which has p facilities, we greedily add x additional facilities. In this heuristic also, we take the value of x as a fraction of total number of facilities p, $x = p * D_{prt}$. From the list of non-facility nodes we select a node whose addition to Sol causes the least increase in the current fitness value and add it to the solution Sol. We add a total of x facilities in this manner. After this, we iteratively remove the additional x facilities one at a time, in a greedy manner. From the $p + x$ facilities in Sol, we select a facility whose removal causes maximum reduction in the objective value and remove that facility from Sol. We continue to remove further facilities from Sol in the same manner and stop when there are a total of p facilities remaining in Sol.

- LH_5**: Remove randomly and add randomly**
  As part of the fifth low level heuristic, from the given solution Sol which has p facilities, we randomly remove x facilities. Similar to other low level heuristics, in this heuristic also we take the value of x as a fraction of total number of facilities p, $x = p * D_{prt}$. To the resulting solution Sol with $p - x$ facilities, we add x new facilities by randomly selecting one facility at a time from the non-facility nodes. This procedure stops when there are a total of p facilities in the solution Sol.

6.3.5 Local search

At the end of every restart of the hyper-heuristic, we apply a local search operation on the best solution found so far, Sol_{best}. As part of the local search, we have performed a 1-1 exchange operation. In the 1-1 exchange, we have followed a best replacement strategy where for each facility i in Sol_{best}, we search for a node from among all the $n - p$ non-facility nodes to replace i that results in maximum reduction in the objective value of Sol_{best} [178]. Once a facility is

replaced from the solution Sol_{best} with a new node, it may get reintroduced into the solution at a later index. For any facility in the solution Sol_{best}, if we are not able to find a replacement node that reduces the objective value, then such a facility remains in the solution. Then, we consider the next facility in Sol_{best} for replacement. The local search is complete when each of the p facilities in Sol_{best} is considered for replacement. If overall best solution remains unchanged during a restart and previous application of local search then local search is not applied.

6.3.6 Selection methodology

There are many selection methodologies that are proposed in the literature such as random selection, random gradient selection, random permutation, random permutation gradient, greedy selection [74]. Random selection method randomly selects one low-level heuristic at each step of the search process. Random gradient selection is an extension of the random selection technique which applies the randomly selected heuristic in a loop until there is no improvement. Random permutation method makes a random ordering of all the available low-level heuristics and in each step of the search operation applies one low-level heuristic in the newly generated order. Random permutation gradient selection is an extension of random permutation selection. Finally, greedy selection is an exhaustive method that applies all the low-level heuristics and the heuristic that produces the best solution among all the low-level heuristics is considered as selected. We have used random selection and greedy selection methods as part of our proposed approach which are suitable for the given less number of low-level heuristics, whereas the other selection mechanisms are useful when the number of low-level heuristics is large [183]. In this chapter, we refer to the hyper-heuristic version with random selection as HH_Rnd and hyper-heuristic version with greedy selection as HH_Grd.

6.3.7 Acceptance criteria

In each step of the search operation, whether to accept the newly generated solution or not is decided based on the acceptance criteria. Several acceptance criterias have been mentioned in the literature [74]. We have experimented with two acceptance criterias namely all acceptance (AA) and only improvement (OI). We obtained better results with the OI criteria which we have reported in the Section 6.4.

6.4 Computational results

We have implemented both our hyper-heuristic based approaches for the RpMF in C and performed all our experiments on a Linux based Intel Core i5 7500 system with 8 GB memory running at 3.40 GHz. Following are the various parameters of the hyper-heuristics and their corresponding values: $N_{rst} = 10$, $N_{itrs} = 10$, $deg_{prt} = 0.5$. We have empirically picked these parameter values after a lot of experimentation. We have evaluated the performance of our two approaches on the same datasets as used in [181] and compared the results obtained with the optimum or best known objective values reported by state-of-the-art methods used for solving the RpMF in [181]. The datasets used in [181] are derived from the base instances used in [141]. The datasets include homogeneous and non-homogeneous instances. In the homogeneous instances each failable facility has equal, uniform probability of failure $q = 0.05$. In the non-homogeneous instances, the probability of failure varies from facility to facility. Within the homogeneous instances, there are two groups of instances namely Type I and Type II. Type I instances are derived from the base instances *49UFLP, 88UFLP, 150UFLP* which contain the number of nodes in the set $\{49, 88, 150\}$, where each node represents a city in the United States. The demand associated with each city is taken proportional to the corresponding city population. The cost of a customer not being served in Type I instances is taken as 10^4. In Type I instances, only the dummy facility to which all unserved customers are assigned is non-failable, and every other facility is considered as failable. Type II instances are derived from the base instances *50EucUFLP, 100EucUFLP* which contain randomly generated datasets with the number of nodes 50 and 100 located in the range of [0, 1]x[0, 1]. The demand associated with each node in the Type II instances is a random number in the range [0, 1000]. Similarly, the cost of a customer not being served in Type II instances is taken as 10.

As part of the experimentation in [181], they have randomly selected subsets of nodes from the original Type I, Type II instances of [141] and generated smaller instances with the number of nodes $n \in \{20, 25, 30, 35, 40, 45\}$. The number of facilities to be located is considered from the set $p \in \{4, 5, 6\}$. For every original instance of the original datasets namely *49UFLP, 50EucUFLP, 88UFLP, 100EucUFLP, 150UFLP* and the set of (n, p) values, three new instances are generated accounting to a total of 270 instances which include 162 Type I and 108 Type II homogeneous instances. Similarly for the non-homogeneous case, three new datasets are generated from each of the original instances *49UFLP, 50EucUFLP, 88UFLP, 100EucUFLP, 150UFLP* with the number of nodes $n \in \{20, 25\}$ and the number of facilities to be located

p is fixed as 4. The value of the failure probability q is randomly taken from 4 different intervals $m \pm s$ with $m \in \{0.03, 0.07\}$ and $s \in \{0.003, 0.025\}$ resulting in a total of 120 non-homogeneous instances which include 60 instances each with the number of nodes 20 and 25.

We have obtained instance by instance results from the first author of [181] and utilized these results for comparisons. For the homogeneous datasets, Table 6.2, and Table 6.3 present the average percentage deviation of our results from the optimum values for Type I, Type II instances respectively. In both these tables, the first column named n provides the number of nodes in the instance. The second column named p gives the number of facilities to be located. The third column named $Dataset$ represents the original dataset from which the instance is derived. The fourth column gives the average of the three percentage deviations obtained on the three instances having the same values of (n, p) by HH_Rnd, whereas the fifth column gives the similar average deviations obtained by HH_Grd. As mentioned earlier, there are three instances for each (n, p) dimension, and we have reported the average deviation of all the three instances of a given (n, p) dimension which are derived from a given original instance. For example, given the original instance 150UFLP and the (n, p) combination of $n = 20$ and $p = 4$, the three generated instances are: *data150UFLP_4_20_1.dat*, *data150UFLP_4_20_2.dat*, *data150UFLP_4_20_3.dat*. We have obtained the objective values of our approach on solving the RpMF on these three instances and found the percentage of deviation from the optimum for each of these three instances. The averages of these three percentage deviations for HH_Rnd and HH_Grd are reported in the fourth and and fifth columns of Table 6.2, and Table 6.3. The sixth, seventh and eighth columns in these two tables report the execution times (in seconds) taken by the CPLEX for model proposed in [181] and our HH_Rnd, HH_Grd variants respectively. Both HH_Rnd and HH_Grd obtained the optimal objective values on 125 of the 162 Type I homogeneous instances. Similarly, we obtained optimal objective values on 101 of the 108 Type II homogeneous instances. Overall, we have obtained the optimal values on 226 out of a total of 270 instances of homogeneous instances combining both the Type I and Type II instances. Though our approaches are executed on a better computer system compared to the one used by Albareda-Sambola et al.[181] (A system with 3.16GHz Intel Core Duo E8500 CPU and 3.4GB RAM Vs a system with 3.40GHz Intel Core i5 7500 CPU and 8 GB RAM used to execute our approaches), it can be observed from the execution times reported in Table 6.2 and Table 6.3 that our approaches take negligible execution times on almost all the instances,

thereby proving computational efficiency of our proposed HH_Rnd and HH_Grd. Between HH_Rnd and HH_Grd, HH_Rnd is faster as expected.

Table 6.2: HH_Grd results on Homogeneous Type I instances

n	*p*	Dataset	$\%Dev_{HH\_Rnd}$	$\%Dev_{HH\_Grd}$	$Time_{CPLEX}$	$Time_{HH\_Rnd}$	$Time_{HH\_Grd}$
		150UFLP	**0.00**	**0.00**	0.34	**0.00**	0.01
	4	49UFLP	**0.00**	**0.00**	0.56	**0.00**	0.03
		88UFLP	0.10	0.10	0.59	**0.00**	0.03
		150UFLP	**0.00**	**0.00**	0.44	**0.01**	0.03
20	5	49UFLP	**0.00**	**0.00**	1.30	**0.01**	0.06
		88UFLP	**0.00**	**0.00**	0.76	**0.01**	0.06
		150UFLP	**0.00**	**0.00**	0.54	**0.01**	0.04
	6	49UFLP	**0.00**	**0.00**	0.91	**0.02**	0.09
		88UFLP	**0.00**	**0.00**	0.94	**0.02**	0.08
		150UFLP	**0.00**	**0.00**	0.70	**0.00**	0.02
	4	49UFLP	0.01	0.01	3.49	**0.01**	0.04
		88UFLP	**0.00**	**0.00**	2.08	**0.01**	0.04
		150UFLP	0.09	0.09	1.09	**0.01**	0.03
25	5	49UFLP	**0.00**	**0.00**	2.65	**0.02**	0.08
		88UFLP	**0.00**	**0.00**	1.65	**0.02**	0.08
		150UFLP	**0.00**	**0.00**	1.15	**0.01**	0.04
	6	49UFLP	0.01	0.01	3.05	**0.03**	0.11
		88UFLP	**0.00**	**0.00**	2.18	**0.02**	0.11
		150UFLP	**0.00**	**0.00**	1.30	**0.01**	0.03
	4	49UFLP	0.01	0.01	6.54	**0.01**	0.05
		88UFLP	**0.00**	**0.00**	3.05	**0.01**	0.05
		150UFLP	0.07	0.07	2.00	**0.01**	0.04
30	5	49UFLP	**0.00**	**0.00**	7.61	**0.02**	0.12
		88UFLP	**0.00**	**0.00**	3.30	**0.02**	0.11
		150UFLP	**0.00**	**0.00**	2.23	**0.01**	0.05
	6	49UFLP	**0.00**	**0.00**	8.29	**0.03**	0.15
		88UFLP	**0.00**	**0.00**	9.13	**0.03**	0.16
		150UFLP	0.04	0.04	2.34	**0.01**	0.03
	4	49UFLP	0.03	0.03	6.19	**0.01**	0.08
		88UFLP	**0.00**	**0.00**	15.60	**0.01**	0.08
		150UFLP	0.07	0.07	3.19	**0.01**	0.07
35	5	49UFLP	0.01	0.01	22.18	**0.04**	0.16
		88UFLP	**0.00**	**0.00**	14.40	**0.03**	0.14
		150UFLP	0.07	0.07	4.13	**0.02**	0.07
	6	49UFLP	**0.00**	**0.00**	33.31	**0.04**	0.20
		88UFLP	**0.00**	**0.00**	13.54	**0.04**	0.22
		150UFLP	0.13	0.13	3.78	**0.01**	0.05
	4	49UFLP	0.05	0.05	31.43	**0.02**	0.10
		88UFLP	**0.00**	**0.00**	10.65	**0.02**	0.09
		150UFLP	0.13	0.13	6.49	**0.01**	0.06
40	5	49UFLP	**0.00**	**0.00**	26.85	**0.04**	0.17
		88UFLP	**0.00**	**0.00**	40.54	**0.04**	0.18
		150UFLP	0.06	0.06	6.35	**0.03**	0.07
	6	49UFLP	**0.00**	**0.00**	51.68	**0.06**	0.24
		88UFLP	**0.00**	**0.00**	26.03	**0.05**	0.24
		150UFLP	0.07	0.07	12.83	**0.01**	0.05
	4	49UFLP	0.01	0.01	127.74	**0.02**	0.10
		88UFLP	0.01	0.01	35.48	**0.02**	0.11
		150UFLP	0.02	0.02	7.65	**0.02**	0.08
45	5	49UFLP	**0.00**	**0.00**	77.54	**0.05**	0.23
		88UFLP	**0.00**	**0.00**	105.20	**0.05**	0.23
		150UFLP	0.07	0.07	11.59	**0.02**	0.10
	6	49UFLP	**0.00**	**0.00**	108.41	**0.07**	0.34
		88UFLP	**0.00**	**0.00**	33.07	**0.07**	0.34

Table 6.3: HH_Grd results on Homogeneous Type II instances

n	*p*	**Dataset**	$\%Dev_{HH\_Rnd}$	$\%Dev_{HH\_Grd}$	$Time_{CPLEX}$	$Time_{HH\_Rnd}$	$Time_{HH\_Grd}$
20	4	100EuclUFLP	**0.00**	**0.00**	0.77	**0.01**	0.03
		50EucUFLP	**0.00**	**0.00**	0.63	**0.01**	0.03
	5	100EuclUFLP	**0.00**	**0.00**	1.94	**0.01**	0.07
		50EucUFLP	**0.00**	**0.00**	1.42	**0.01**	0.07
	6	100EuclUFLP	0.05	**0.00**	1.29	**0.02**	0.10
		50EucUFLP	**0.00**	**0.00**	2.73	**0.02**	0.11
25	4	100EuclUFLP	**0.00**	**0.00**	3.62	**0.01**	0.05
		50EucUFLP	**0.00**	**0.00**	9.51	**0.01**	0.05
	5	100EuclUFLP	**0.00**	**0.00**	8.24	**0.02**	0.10
		50EucUFLP	**0.00**	**0.00**	9.71	**0.02**	0.11
	6	100EuclUFLP	**0.00**	**0.00**	3.33	**0.03**	0.14
		50EucUFLP	**0.00**	**0.00**	7.63	**0.03**	0.15
30	4	100EuclUFLP	**0.00**	**0.00**	22.57	**0.01**	0.06
		50EucUFLP	**0.00**	**0.00**	13.84	**0.01**	0.06
	5	100EuclUFLP	**0.00**	**0.00**	22.99	**0.02**	0.13
		50EucUFLP	**0.00**	**0.00**	14.48	**0.02**	0.13
	6	100EuclUFLP	**0.00**	**0.00**	34.80	**0.03**	0.18
		50EucUFLP	**0.00**	**0.00**	15.56	**0.04**	0.19
35	4	100EuclUFLP	**0.00**	**0.00**	26.45	**0.01**	0.08
		50EucUFLP	**0.00**	**0.00**	58.58	**0.01**	0.08
	5	100EuclUFLP	**0.00**	**0.00**	66.86	**0.03**	0.16
		50EucUFLP	0.01	0.01	74.94	**0.03**	0.18
	6	100EuclUFLP	**0.00**	**0.00**	34.42	**0.04**	0.23
		50EucUFLP	0.01	0.01	81.11	**0.05**	0.25
40	4	100EuclUFLP	**0.00**	**0.00**	40.59	**0.02**	0.11
		50EucUFLP	**0.00**	**0.00**	154.16	**0.02**	0.11
	5	100EuclUFLP	**0.00**	**0.00**	51.39	**0.04**	0.20
		50EucUFLP	**0.00**	**0.00**	284.95	**0.04**	0.21
	6	100EuclUFLP	**0.00**	**0.00**	32.13	**0.06**	0.28
		50EucUFLP	**0.00**	**0.00**	156.18	**0.06**	0.29
45	4	100EuclUFLP	**0.00**	**0.00**	134.06	**0.02**	0.12
		50EucUFLP	0.02	0.02	80.35	**0.02**	0.13
	5	100EuclUFLP	**0.00**	**0.00**	115.33	**0.05**	0.25
		50EucUFLP	**0.00**	**0.00**	269.34	**0.05**	0.25
	6	100EuclUFLP	**0.00**	**0.00**	167.16	**0.07**	0.35
		50EucUFLP	**0.00**	**0.00**	395.12	**0.07**	0.35

Apart from newly generated Type I, Type II homogeneous instances, Albareda-Sambola et al. [181] have also experimented on large instances which are introduced by Snyder and Daskin [141]. We have obtained these large instances from the repository provided by [141] and reported the results obtained by our HH_Rnd and HH_Grd methods on these large instances in the Table 6.4. For each of these 15 large instances, we generated the objective values using the solutions which are provided by [181] and compared the solutions and objective values obtained by our approach. In Table 6.4, first column gives the name of the dataset, second column, n, is the number of nodes in the instance, third column, p, gives the number of facilities

to be located and the fourth, fifth columns give the percentage deviation of the results of our proposed HH_Rnd and HH_Grd methods with respect to the best known solution reported by [181] on the same instance. Columns six, seven and eight present the execution times (in seconds) of CPLEX, HH_Rnd and HH_Grd respectively on each instance. As can be seen from fourth and fifth columns in Table 6.4, out of the 15 instances, HH_Rnd obtained the same or better objective values on 13 instances, whereas HH_Grd obtained the same or better objective values on all the 15 instances when compared to the objective values of the best known solutions reported by [181]. Compared to the proposed HH_Rnd, HH_Grd obtained better objective values on 3 large instances. Both our proposed HH_Rnd, HH_Grd approaches take much less execution times in comparison to the $CPLEX$, while HH_Rnd is the faster approach between the two proposed approaches.

Table 6.4: HH_Grd results on Large Homogeneous instances

Dataset	***n***	***p***	$\%Dev_{HH\_Rnd}$	$\%Dev_{HH\_Grd}$	$Time_{CPLEX}$	$Time_{HH\_Rnd}$	$Time_{HH\_Grd}$
		5	**0.00**	**0.00**	94.33	**0.06**	0.23
49UFLP	49	10	**0.00**	**0.00**	90.44	**0.35**	1.51
		20	**0.00**	**0.00**	28.80	**4.06**	17.39
		5	**-0.03**	**-0.03**	174.00	**0.06**	0.26
50EucUFLP	50	10	**-0.01**	**-0.01**	59.23	**0.40**	1.87
		20	**-0.01**	**-0.01**	26.94	**4.81**	19.82
		5	**0.00**	**0.00**	7200.00	**0.18**	0.73
88UFLP	88	10	**-0.30**	**-0.30**	7200.00	**1.18**	4.44
		20	**0.00**	**-0.01**	3522.53	**12.67**	51.50
		5	0.69	**0.00**	7200.00	**0.20**	1.02
100EucUFLP	100	10	**-0.19**	**-0.19**	7200.00	**1.39**	6.40
		20	**-0.67**	**-0.67**	7200.00	**17.65**	75.81
		5	**0.00**	**0.00**	262.83	**0.20**	0.74
150UFLP	150	10	**0.00**	**0.00**	2044.42	**0.52**	1.97
		20	0.14	**0.00**	1131.13	**1.96**	8.39

For the non-homogeneous datasets with the number of nodes, $n = 20$, the authors of [181] have provided the optimum values for each instance. Whereas for the non-homogeneous datasets with the number of nodes, $n = 25$, they have provided best known solution objective values. In the Table 6.5 and the Table 6.6, we have presented the percentage deviation of our results from the optimum and best known solution objective values for the instances with the number of nodes 20, 25 respectively. In both these tables, first column, $Dataset$, represents the original dataset from which the instances are generated. Columns 2, 3 give the average of percentage deviations of all the three instances generated from the original instance having the probability of failure $q \in 0.03 \pm 0.003$, which are shown under the column title A that represents the range of probability of failure. Columns 4, 5 give the results on instances having the probability of

failure $q \in 0.03 \pm 0.025$, and shown under the column title B. Similarly, columns 6, 7 give the results on instances having the probability of failure $q \in 0.07 \pm 0.003$, and shown under the column title C. Finally, columns 8, 9 give the results on instances having the probability of failure $q \in 0.07 \pm 0.025$, that are shown under the column title D. On the non-homogeneous datasets with 20 nodes and 4 facilities, both our proposed HH_Rnd and HH_Grd achieved the optimum value on 45 out of 60 instances. Similarly, on the non-homogeneous datasets with 25 nodes and 4 facilities, out of 60 instances HH_Grd achieved the best known solution or improved the solution quality on 43 instances, while HH_Rnd achieved the best known solution or improved the solution quality on 42 instances. Overall, HH_Grd obtained equal or improved quality solutions as compared to the best known solutions on a total of 88 instances out of a total of 120 non-homogeneous instances, while HH_Rnd obtained equal or improved quality solutions as compared to the best known solutions on 87 instances. Table 6.7 presents the medians of execution times (in seconds) of the proposed approaches HH_Rnd and HH_Grd with respect to the CPLEX times from [181] for the non-homogeneous datasets. In Table 6.7, 1st column gives the ranges of probability of failure. Columns 2, 3 and 4 present the times of CPLEX, HH_Rnd and HH_Grd for datasets with 20 nodes, while columns 5, 6, 7 present the execution times (in seconds) of CPLEX, HH_Rnd and HH_Grd for datasets with 25 nodes.

Table 6.5: HH_Grd results on Non-Homogenous instances with n = 20 nodes and p = 4 facilities

	A		B		C		D	
Dataset	$\%Dev_{HH\_Rnd}$	$\%Dev_{HH\_Grd}$	$\%Dev_{HH\_Rnd}$	$\%Dev_{HH\_Grd}$	$\%Dev_{HH\_Rnd}$	$\%Dev_{HH\_Grd}$	$\%Dev_{HH\_Rnd}$	$\%Dev_{HH\_Grd}$
49UFLP	0.20	0.20	0.03	0.03	**0.00**	**0.00**	**0.00**	**0.00**
50EucUFLP	**0.00**	**0.00**	**0.00**	**0.00**	0.02	0.02	**0.00**	**0.00**
88UFLP	0.43	0.43	0.22	0.22	0.18	0.18	0.50	0.50
100EuclUFLP	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
150UFLP	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

Table 6.6: HH_Grd results on Non-Homogenous instances with n = 25 nodes and p = 4 facilities

	A		B		C		D	
Dataset	$\%Dev_{HH\_Rnd}$	$\%Dev_{HH\_Grd}$	$\%Dev_{HH\_Rnd}$	$\%Dev_{HH\_Grd}$	$\%Dev_{HH\_Rnd}$	$\%Dev_{HH\_Grd}$	$\%Dev_{HH\_Rnd}$	$\%Dev_{HH\_Grd}$
49UFLP	0.22	0.22	0.16	0.16	**-1.40**	**-1.40**	**0.00**	**0.00**
50EucUFLP	**-0.01**	**-0.01**	**0.00**	**0.00**	**0.00**	**0.00**	0.01	0.01
88UFLP	0.32	0.32	0.08	0.08	**-1.94**	**-1.94**	**-1.55**	**-1.55**
100EuclUFLP	**0.00**	**0.00**	**-0.02**	**-0.02**	**0.00**	**0.00**	0.02	0.02
150UFLP	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

Table 6.7: Average CPU times for the Non-Homogenous instances

	$n = 20$			$n = 25$		
	$Time_{CPLEX}$	$Time_{HH\_Rnd}$	$Time_{HH\_Grd}$	$Time_{CPLEX}$	$Time_{HH\_Rnd}$	$Time_{HH\_Grd}$
$q \in 0.03 \pm 0.003$	7.39	**0.01**	0.02	48	**0.01**	0.02
$q \in 0.03 \pm 0.025$	4.78	**0.01**	0.02	29.49	**0.01**	0.02
$q \in 0.07 \pm 0.003$	112.19	**0.01**	0.02	1255.88	**0.01**	0.02
$q \in 0.07 \pm 0.025$	122.92	**0.01**	0.02	436.06	**0.01**	0.02

After taking the averages of the deviations of 3 instances with the same (n, p) dimension, we can observe from the Table 6.2, on Type I homogeneous instances both our proposed HH_Rnd and HH_Grd obtained the optimal objective values on 34 out of 54 different (n, p) dimension datasets. In the 20 cases of the Type I homogeneous instances where our approaches did not achieve the optimal values, the average percentage deviations are very small which are in the range of 0.01% to 0.13%. And on Type II homogeneous instances, HH_Grd obtained the optimal values on 33 cases and HH_Rnd obtained the optimal values on 32 cases out of the total 36 different (n, p) dimension datasets as can be seen in Table 6.3. On the instances with $n = 20$ and $p = 6$ which are derived from original dataset 100EuclUFLP, HH_Rnd has an average deviation of 0.05% from the optimum but we obtained optimal values using the HH_Grd on these same instances as reported in the 5th row of Table 6.3. In those cases of Type II homogeneous instances where we did not achieve the optimal values, the deviations are very small which are in the range of 0.01% to 0.05%. Similarly in the non-homogeneous case, for the datasets with 20 nodes and 4 facilities, looking at averages of percentage deviations of 3 instances having the failure probability belonging to the same interval of $q \in m \pm s$, we can observe from Table 6.5 that both our approaches HH_Rnd and HH_Grd obtained the optimal values on 13 out of 20 datasets with different failure probability values $q \in m \pm s$. In the 7 cases that our approaches didn't obtain the optimal values in Table 6.5 the deviations are small which vary in the range of 0.02% to 0.5%. Also, for the datasets with 25 nodes and 4 facilities, looking at averages of deviations in Table 6.6, we can observe that on 14 out of 20 datasets both our approaches HH_Rnd and HH_Grd obtained or improved upon the best known solution. In the 6 cases that our approaches didn't achieve the best known solutions in Table 6.6 the deviations are small which vary in the range of 0.01% to 0.32%. As can be observed from Table 6.7, for all the four failure probability ranges of the non-homogeneous datasets, our approaches take only negligible execution times to generate the results.

6.5 Conclusions

In this chapter, we have proposed two multi-start hyper-heuristic approaches based on greedy selection and random selection mechanisms for the reliable p-median problem with at facility service. We have evaluated our approaches on homogeneous datasets which have an equal uniform failure probability for all the failable facilities as well as on non-homogeneous datasets that have facility dependent failure probabilities. The results of the proposed approaches are compared with the state-of-the-art approaches available in literature for RpMF. Out of the total 405 instances, our greedy selection based hyper-heuristic achieved the optimal solutions or improved the best known solutions on 329 instances, while our random selection based hyper-heuristic achieved the optimal solutions or improved the best known solutions on 326 instances. In all the cases, our proposed approaches obtained solutions of good quality in negligible execution times proving their computational efficiency.

Chapter 7

Conclusions and directions for future research

In this thesis, we have developed heuristic approaches for six facility location problems which are all $\mathcal{NP}$-hard problems. We have used two evolutionary approaches, namely discrete differential evolution (DDE) and genetic algorithm (GA), and hyper-heuristic approaches to solve the considered facility location problems. While developing our evolutionary approaches and hyper-heuristics, we have incorporated the problem-specific knowledge wherever possible in the solution encodings, methods of generating initial solutions, genetic operators and local search methods for the considered problems. Devising these approaches which have performed as good as or better than the state-of-the-art approaches on each problem constitutes the major contribution of this thesis.

The contributions made by our work in each chapter along with possible directions for future research are described in the following.

In Chapter 2, we have proposed a population based evolutionary approach, namely the discrete differential evolution algorithm for the ACLP. To represent each solution in the population, we have used a simple bit vector of length equal to the number of nodes in the network. We have used a semi greedy approach to generate the population of initial solutions. Given the binary nature of solutions, as part of mutation we have flipped binary value associated with each location of the best solution according to the given mutation probability. We have deployed a simple uniform crossover. We have implemented a repair operation to check for the feasibility of the resulting solution after crossover/mutation and to make it feasible if it is not feasible and also to further improve its fitness. Apart from the benchmark instances used in

[1, 83], we have also evaluated our approach on larger instances which we have derived from Beasley's OR-library[1] and TSPLIB[2]. Computational results show that on most of the instances, our approach performed as good as or better than the state-of-the-art approaches for ACLP [1, 83]. The results of the statistical significance test also prove that the improvement achieved with our approach is significant and it is due to the algorithmic merit. We have not reported the comparison of execution times as both our proposed DDE approach and the ACO based approach are executed for the same amount of time on each instance. Our DDE based approach for the unweighted ACLP can be extended for solving other variants of ACLP.

In Chapter 3, we have worked on two variants of anti-covering location problem, viz. disruptive anti-covering location problem (DACLP) and weighted anti-covering location problem (WACLP). In this chapter, we have proposed two population based metaheuristics for the considered problems. As our first approach, we have extended our discrete differential evolution based approach for the ACLP to both DACLP and WACLP, and as our second approach, we have developed a genetic algorithm based method for solving DACLP and WACLP. Though both differential evolution and genetic algorithm belong to the broad class of evolutionary algorithms and make use of crossover and mutation, we have used completely different solution encodings and devised the crossover and mutation operations quite differently for the two proposed approaches. In our DDE based approach for DACLP and WACLP, we have used the solution representation, mutation, crossover and repair operations just like in the DDE approach for ACLP while incorporating the problem specific information. In our proposed GA approach for DACLP and WACLP, we have represented the solution as an ordered set of locations and designed the mutation and crossover operations accordingly. We have used probabilistic binary tournament selection method to choose the two parents for crossover and a single parent for mutation in the GA approach. It was proven that the binary tournament selection approach outperforms the roulette wheel selection method while also being computationally less expensive. We have applied crossover, mutation operations in a mutually exclusive manner and if the fitness of the resulting solution obtained after crossover/mutation is within a certain percentage of the best solution's fitness, we have also performed a local search to further improve its fitness. Ours is a steady-state GA that generates a single child solution in each generation that is considered for replacement. We have evaluated the performance of our approaches for DACLP and WACLP on the 80 ACLP instances with upto 1577 nodes which are introduced in Chapter 2 and are

[1] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/esteininfo.html
[2] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html

modified to have node weights while solving for WACLP. Computational results show the effectiveness of our approaches in solving both the considered ACLP variants. Our approaches are the first metaheuristic approaches for the DACLP. Our metaheuristic approaches will serve as motivation to other researchers to develop new metaheuristic approaches for the DACLP and WACLP. Similar approaches can be developed for other related problems such as dominating set, independent set, and vertex cover variants.

In Chapter 4, we have solved the OCMCLP by using a genetic algorithm based approach. We have evaluated the performance of our proposed approach on benchmark instances of the problem and compared the results with two interchange heuristics available in the literature for OCMCLP. In this approach, we have represented a solution as an ordered set of locations at which facilities can be located. We have utilized the problem specific knowledge in each of the operations like initial solution generation, crossover, mutation and local search. On most of the instances, our GA based approach has obtained solutions of superior quality in comparison to the existing methods. However, our approach needs more execution time than these methods. Our GA based approach is the maiden metaheuristic approach that has been developed for OCMCLP. Averbakh et al. [2] mentioned that even though they have implemented a tabu search approach and a variable neighborhood search approach, they did not present their results in [2] because of the observation that there is no significant improvement in the solution quality by these two metaheuristic approaches over the solution quality obtained by the interchange heuristics. Hence, population based metaheuristics appear to be better suited for this problem.

Future research could build other metaheuristics techniques for the OCMCLP and can be compared with our GA approach and two interchange heuristics of [2]. Ideas presented here can be used in developing other metaheuristic approaches for the OCMCLP or for other similar problems under cooperative coverage model. Similar approaches can be developed for other facility location problems also where facilities can be located along the edges.

In Chapter 5, we have proposed a hyper-heuristic based approach with naive Bayes classifier for the reliability p-median problem. We have compared the results of our approach with the state-of-art approaches available in the literature [142]. Two methods of initial solution generation are used in this approach, the first is a randomized greedy approach for the first iteration of the hyper-heuristic and in all the other restarts, we have employed a random approach of generating the initial solutions. We have implemented four low level heuristics each of which generates a feasible solution for the problem and also implemented a local-search operation to further improve the quality of the best solution. We have made use of the naive Bayes classifier

to skip executing one of the four low level heuristics in latter iterations. The effectiveness of our approach can be observed in terms of the solution quality.

In Chapter 6, we have proposed two multi-start hyper-heuristic approaches based on greedy selection and random selection for the reliable p-median problem with at facility service. We have evaluated our approaches on homogeneous datasets as well as on non-homogeneous datasets. When compared with the state-of-the-art approaches available in literature for RpMF, our greedy selection based hyper-heuristic achieved the optimal solutions or improved the best known solutions on 329 out of the total 405 instances, whereas our random selection based hyper-heuristic achieved the optimal solutions or improved the best known solutions on 326 out of the total 405 instances. Using the proposed approaches, we could obtain solutions of good quality in negligible execution times which proves the efficacy of our approaches.

Our hyper-heuristic approaches are the first hyper-heuristic approaches for solving RpMP and RpMF. While ours are greedy/random selection based hyper-heuristic approaches, these will serve as a motivation for other researchers to develop other kind of hyper-heuristic approaches such as generational hyper-heuristic approaches and those based on other selection mechanisms for RpMP, RpMF and other facility location problems. No approach exists in the literature so far that combines a machine learning technique with a hyper-heuristic approach for solving the facility location problems. Our approach for RpMP is the first approach that combines a hyper-heuristic method with a machine learning technique for a facility location problem. This paves the way for other researchers to develop analogous methods using other machine learning techniques including but not limited to naive Bayes classifier, support vector machine (SVM), reinforcement learning, regression etc. for solving the different variety of facility location problems.

In all, six facility location problems have been considered in this thesis. Chapter 2 and Chapter 3 are concerned with anti-covering location problems, whereas Chapter 4 deals with facility location problem under cooperative coverage model. The last two chapters, namely Chapter 5 and Chapter 6 are focused on facility location problems that take facility failures into consideration. The approaches developed for solving these three broad categories of facility location models considered in this thesis help in gaining useful insights on how to proceed when solving such problems using heuristic approaches. This knowledge can be utilized while solving other facility location problems or other similar discrete optimization problems.

The field of facility location problems is an active area of research owing to practical applications. Not only new and better methods are developed on a regular basis for existing

problems, but also new facility location problems continue to emerge as a result of progress in technology and human civilization. These new and better methods are developed either by discovering and utilizing new problem characteristics or using the already explored problem characteristics in a new manner or a combination thereof. Sometimes the insights gained while solving a new problem lead to the development of better methods for existing problems. Hence, the knowledge in the field of facility location problems is continuously advancing. So, there remains a possibility of the development of better heuristic approaches in future for the facility location problems considered here.

References

[1] S. S. CHAUDHRY. **A genetic algorithm approach to solving the anti-covering location problem**. *Expert Systems*, **23**(5):251–257, 2006. (xv, 23, 28, 29, 30, 33, 34, 123)

[2] I. AVERBAKH, O. BERMAN, D. KRASS, J. KALCSICS, AND S. NICKEL. **Cooperative covering problems on networks**. *Networks*, **63**(4):334–349, 2014. (xv, xvi, 4, 60, 61, 62, 63, 65, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 124)

[3] M.S DASKIN AND L.K DEAN. **Location of health care facilities**. In *Operations research and health care*, pages 43–76. Springer, 2005. (2, 4)

[4] R. CHURCH AND C.R. VELLE. **The maximal covering location problem**. *Papers of the Regional Science Association*, **32**(1):101–118, 1974. (2, 4, 60)

[5] V. MARIANOV AND D. SERRA. *4 Location problems in the public sector:In facility location:applications and theory*. Springer, 2002. (2)

[6] L. COOPER. **Location-allocation problems**. *Operations research*, **11**(3):331–343, 1963. (2)

[7] F. PLASTRIA. **Continuous location problems: research, results and questions**. *Facility location: a survey of applications and methods*, pages 85–127, 1995. (3)

[8] Z. DREZNER. **Continuous Facility Location Problems**. In *The Palgrave Handbook of Operations Research*, pages 269–306. Springer, 2022. (3)

[9] T. A. HARTMANN, S. LENDL, AND G. J. WOEGINGER. **Continuous facility location on graphs**. *Mathematical Programming*, **192**(1):207–227, 2022. (3)

[10] J. BRIMBERG, P. HANSEN, N. MLADENOVIĆ, AND S. SALHI. **A survey of solution methods for the continuous location-allocation problem**. *International journal of operations research*, **5**(1):1–12, 2008. (3, 5)

[11] Z. ULUKAN AND E. DEMIRCIOĞLU. **A Survey of discrete facility location problems**. *World academy of science, engineering and technology, international journal of social, behavioral, educational, economic, business and industrial engineering*, **9**(7):2450–2455, 2015. (3)

[12] Z. DREZNER. *Facility location:A survey of applications and methods*. Springer, 1995. (3, 4)

[13] S. BASU, M. SHARMA, AND P.S. GHOSH. **Metaheuristic applications on discrete facility location problems: a survey**. *OPSEARCH*, **52**(3):530–561, 2015. (3, 5)

[14] J. CURRENT, M. DASKIN, AND D. SCHILLING. **3 Discrete network location models**. *Facility location applications and theory*, pages 81–118, 2004. (3)

[15] B.C. TANSEL, R.L. FRANCIS, AND T.J. LOWE. **State of the art - location on networks: a survey. Part I: the p-center and p-median problems**. *Management science*, **29**(4):482–497, 1983. (3)

[16] A. AHMADI-JAVID, P. SEYEDI, AND S. S. SYAM. **A survey of healthcare facility location**. *Computers & Operations Research*, **79**:223–263, 2017. (4)

[17] R.Z. FARAHANI, M. STEADIESEIFI, AND N. ASGARI. **Multiple criteria facility location problems: A survey**. *Applied mathematical modelling*, **34**(7):1689–1709, 2010. (4)

[18] O. BERMAN, Z. DREZNER, AND D. KRASS. **Cooperative cover location problems: The planar case**. *IIE Transactions*, **42**(3):232–246, 2009. (4, 60)

[19] F.R. ZANJIRANI, A. NASRIN, H. NOOSHIN, H. MAHTAB, AND M. GOH. **Covering problems in facility location: A review**. *Computers & industrial engineering*, **62**(1):368–407, 2012. (4)

[20] O. BERMAN, Z. DREZNER, AND D. KRASS. **Generalized coverage: new developments in covering location models**. *Computers & operations research*, **37**(10):1675–1687, 2010. (4)

[21] F. PLASTRIA. **Continuous covering location problems**. *Facility location: applications and theory*, **1**:37–79, 2002. (4)

[22] J. CURRENT AND M. ÓKELLY. **Locating emergency warning sirens**. *Decision sciences*, **23**(1):221–234, 1992. (4)

[23] O. BERMAN, Z. DREZNER, AND D. KRASS. **Discrete Cooperative Covering Problems**. *J Oper Res Soc*, **62**(11):2002–2012, Nov 2011. (4)

[24] O. BERMAN. **The p maximal cover - p partial center problem on networks**. *European Journal of Operational Research*, **72**(2):432 – 442, 1994. (4)

[25] P. CAPPANERA, G. GALLO, AND F. MAFFIOLI. **Discrete facility location and routing of obnoxious activities**. *Discrete applied mathematics*, **133**(1):3–28, 2003. (4)

[26] E. MELACHRINOUDIS. **The location of undesirable facilities**. In *Foundations of location analysis*, pages 207–239. Springer, 2011. (4, 61)

[27] Z. DREZNER, P. KALCZYNSKI, AND S. SALHI. **The planar multiple obnoxious facilities location problem: A Voronoi based heuristic**. *Omega*, **87**:105–116, 2019. (4, 61)

[28] A.B. ARABANI AND R.Z. FARAHANI. **Facility location dynamics: An overview of classifications and applications**. *Computers & industrial engineering*, **62**(1):408–420, 2012. (4)

[29] R.Z. FARAHANI, M. STEADIESEIFI, AND N. ASGARI. **Multiple criteria facility location problems: A survey**. *Applied mathematical modelling*, **34**(7):1689–1709, 2010. (4)

[30] V. VERTER AND A.E. MURAT. **S. Nickel and J. Puerto: Location theory: a unified approach**. *Mathematical methods of operations research*, **66**(2):369–371, 2007. (4)

[31] C.S. REVELLE AND H.A. EISELT. **Location analysis: a synthesis and survey**. *European journal of operational research*, **165**(1):1–19, 2005. (4)

[32] D.B. SHMOYS, É. TARDOS, AND K. AARDAL. **Approximation algorithms for facility location problems**. In *Proceedings of the twenty-ninth annual ACM symposium on theory of computing*, pages 265–274. ACM, 1997. (4)

[33] D. CELIK TURKOGLU AND M. EROL GENEVOIS. **A comparative survey of service facility location problems**. *Annals of Operations Research*, **292**(1):399–468, 2020. (4)

[34] M.S. DASKIN. *Network and discrete location: models, algorithms, and applications*. John wiley & sons, 2011. (4)

[35] Z. DREZNER AND H. W. HAMACHER. *Facility location: applications and theory*. Springer Science & Business Media, 2004. (4)

[36] R.F. LOVE, J.J.G. MORRIS, AND G.O. WESOLOWSKY. *Facilities location: models & methods*. Publications in operations research. North-Holland, 1988. (4)

[37] A. GHOSH AND G. RUSHTON. *Spatial analysis and location-allocation models*. Van nostrand reinhold, 1987. (4)

[38] P. HANSEN, J. HENDERSON, M. LABBÉ, J. PEETERS, AND J F THISSE. *Systems of cities and facility location*. Taylor & francis, 2013. (4)

[39] J.F THISSE AND H.ZOLLER. *Locational analysis of public facilities*. Studies in mathematical and managerial economics. Elsevier, 1983. (4)

[40] G. LAPORTE, S. NICKEL, AND F. SALDANHA-DA GAMA. **Introduction to location science**. In *Location science*, pages 1–21. Springer, 2019. (4)

[41] L. COOPER. **Heuristic methods for location-allocation problems**. *Siam review*, **6**(1):37–53, 1964. (5)

[42] S.K. JACOBSEN. **Heuristics for the capacitated plant location model**. *European journal of operational research*, **12**(3):253–261, 1983. (5)

[43] B. JAYALAKSHMI AND A. SINGH. **A hybrid artificial bee colony algorithm for the cooperative maximum covering location problem**. *International Journal of Machine Learning and Cybernetics*, **8**(2):691–697, 2017. (5, 60, 61, 68)

[44] B. JAYALAKSHMI AND A. SINGH. **A swarm intelligence approach for the p-median problem**. *International Journal of Metaheuristics*, **5**(2):136–155, 2016. (5, 83)

[45] J.H. HOLLAND. *Adaptation in natural and artificial systems: An introductory analysis with applications in biology, control and artificial intelligence*. University of Michigan Press, Ann Arbor, MI, 1975. (5, 9, 11)

[46] D. GOLDBERG. *Genetic algorithm in search, optimization and machine learning*. Reading, MA :Addison-Wesley, 1989. (5)

[47] R. STORN AND K. PRICE. **Differential evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces**. *Journal of Global Optimization*, **11**:341–359, 1997. (5, 14)

[48] F. GLOVER. **Tabu search - part 1**. *ORSA Journal on Computing*, **1**:190–206, 1989. (5)

[49] F. GLOVER. **Tabu search - part 2**. *ORSA Journal on Computing*, **2**:4–32, 1990. (5)

[50] M. DORIGO, V. MANIEZZO, AND A. COLORNI. **Positive feedback as a search strategy**, 1991. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy. (5)

[51] M. DORIGO, V. O. MANIEZZO, AND A. COLORNI. **Ant system: Optimization by a colony of cooperating agents**. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, **26**:29–41, 1996. (5)

[52] N. MLADENOVIĆ AND P. HANSEN. **Variable neighborhood search**. *Computers & operations research*, **24**(11):1097–1100, 1997. (5)

[53] P. HANSEN AND N. MLADENOVIĆ. **Variable neighborhood search: Principles and applications**. *European journal of operational research*, **130**(3):449–467, 2001. (5)

[54] D. KARABOGA. **An idea based on honey bee swarm for numerical optimization**, 2005. Computer Engineering Department, Erciyes University, Turkey. (5)

[55] M.A. AROSTEGUI, S.N. KADIPASAOGLU, AND B.M. KHUMAWALA. **An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems**. *International journal of production economics*, **103**(2):742–754, 2006. (5)

[56] N. MLADENOVIĆ, J. BRIMBERG, P. HANSEN, AND J.A. MORENO-PÉREZ. **The p-median problem: a survey of metaheuristic approaches**. *European journal of operational research*, **179**(3):927–939, 2007. (5)

[57] M.G.C. RESENDE AND R.F. WERNECK. **A hybrid multistart heuristic for the uncapacitated facility location problem**. *European journal of operational research*, **174**(1):54–68, 2006. (5)

[58] R.C. MARTÍ, P.M. PARDALOS, AND M.G.C. RESENDE. *Handbook of heuristics*. Springer, 2018. (6)

[59] R.R. SHARAPOV. *Genetic algorithms: basic ideas, variants and analysis*. IntechOpen, 2007. (6)

[60] T. BLICKLE AND L. THIELE. **A mathematical analysis of tournament selection**. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, **95**, pages 9–15. Citeseer, 1995. (6)

[61] J.D. SCHAFFER, D. WHITLEY, AND L.J. ESHELMAN. **Combinations of genetic algorithms and neural networks: A survey of the state of the art**. In *[Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 1–37. IEEE, 1992. (9)

[62] M. MITCHELL. *An introduction to genetic algorithms*. Bradford Books, 1998. (9)

[63] J.E. BAKER. **Reducing bias and inefficiency in the selection algorithm**. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21, 1987. (9, 10)

[64] D.E. GOLDBERG AND K. DEB. **A comparative analysis of selection schemes used in genetic algorithms**. In *Foundations of Gentic Algorithms*, pages 69–93. Morgan Kaufmann, 1990. (9, 11)

[65] G. SYSWERDA. **Uniform crossover in genetic algorithms**. In *Proceedings of the Third International Conference on Genetic Algorithms*, **3**, pages 2–9. Morgan Kaufmann, 1989. (12)

[66] L. DAVIS. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991. (13, 39, 51)

[67] S. DAS AND P. N. SUGANTHAN. **Differential evolution: a survey of the state-of-the-art**. *IEEE Transactions on Evolutionary Computation*, **15**(1):4–31, 2011. (14)

[68] R. MALLIPEDDI, P.N. SUGANTHAN, Q. K. PAN, AND M.F. TASGETIREN. **Differential evolution algorithm with ensemble of parameters and mutation strategies**. *Applied soft computing*, **11**(2):1679–1696, 2011. (15)

[69] M. F. TASGETIREN, Q. K. PAN, Y. C. LIANG, AND P.N. SUGANTHAN. **A discrete differential evolution algorithm for the total earliness and tardiness penalties with a common due date on a single-machine**. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (SCIS'07)*, pages 271–278. IEEE, 2007. (15)

[70] Q. K. PAN, M. F. TASGETIREN, AND Y. C. LIANG. **A discrete differential evolution algorithm for the permutation flowshop scheduling problem**. *Computers & Industrial Engineering*, **55**(4):795–816, 2008. (15)

[71] M. F. TASGETIREN, Q.K. PAN, P.N. SUGANTHAN, AND Y. C. LIANG. **A discrete differential evolution algorithm for the no-wait flowshop scheduling problem with total flowtime criterion**. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (SCIS'07)*, pages 251–258. IEEE, 2007. (15, 25)

[72] H. FISHER. **Probabilistic learning combinations of local job-shop scheduling rules**. *Industrial scheduling*, pages 225–251, 1963. (16)

[73] WALLACE B. S. C., FRED W. G., GERALD L. T., AND JOHN D. T. **Probabilistic and parametric learning combinations of local job shop scheduling rules**. *Technical Report, Research Memorandum*, **No. 117**, 1963. (16)

[74] E. K. BURKE, M. GENDREAU, M. HYDE, G. KENDALL, G. OCHOA, E. ÖZCAN, AND R. QU. **Hyper-heuristics: A survey of the state of the art**. *Journal of the Operational Research Society*, **64**(12):1695–1724, 2013. (16, 94, 113)

[75] K. CHAKHLEVITCH AND P. COWLING. **Hyperheuristics: recent developments**. In *Adaptive and multilevel metaheuristics*, pages 3–29. Springer, 2008. (17)

[76] J. DENZINGER, M. FUCHS, AND M. FUCHS. **High Performance ATP Systems by Combining Several AI Methods**. In *Proceedings of the 15th International Joint Conference on Artifical Intelligence - Volume 1*, IJCAI'97, pages 102–107, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. (17)

[77] P. COWLING, G. KENDALL, AND E. SOUBEIGA. **A hyperheuristic approach to scheduling a sales summit**. In *Proceedings of the international conference on the practice and theory of automated timetabling*, pages 176–190. Springer, 2000. (17)

[78] E. CARRIZOSA AND B. G. TÓTH. **Anti-covering problems**. In *Location Science*, pages 115–132. Springer, 2015. (22)

[79] I. D. MOON AND S. S. CHAUDHRY. **An analysis of network location problems with distance constraints**. *Management Science*, **30**(3):290–307, 1984. (22, 23, 34, 35, 48, 49)

[80] M.R. GAREY AND D.S. JOHNSON. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1979. (22, 48)

[81] A. T. MURRAY AND R. L. CHURCH. **Solving the anti-covering location problem using Lagrangian relaxation**. *Computers & Operations Research*, **24**(2):127–140, 1997. (22, 23, 24, 34, 49)

[82] B. DIMITRIJEVIĆ, D. TEODOROVIĆ, V. SIMIĆ, AND M. ŠELMIĆ. **Bee colony optimization approach to solving the anticovering location problem**. *Journal of Computing in Civil Engineering*, **26**(6):759–768, 2011. (23, 34)

[83] P. R. KHORJUVENKAR AND A. SINGH. **A Hybrid Swarm Intelligence Approach for Anti-Covering Location Problem**. In *Proceedings of the 2019 IEEE International Conference on Innovations in Power and Advanced Computing Technologies (i-PACT 2019)*, **1**, pages 1–6. IEEE, 2019. (23, 28, 29, 33, 34, 123)

[84] F. WILCOXON, S. K. KATTI, AND R. A. WILCOX. **Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test**. *Selected tables in mathematical statistics*, **1**:171–259, 1970. (30, 57, 81)

[85] M. R. NIBLETT AND R. L. CHURCH. **The disruptive anti-covering location problem**. *European Journal of Operational Research*, **247**(3):764–773, 2015. (34, 35, 36)

[86] S. S. CHAUDHRY, S. T. MCCORMICK, AND I. D. MOON. **Locating independent facilities with maximum weight: Greedy heuristics**. *Omega*, **14**(5):383–389, 1986. (34, 35, 48, 53)

[87] G. SRIVASTAVA, A. SINGH, AND R. MALLIPEDDI. **A Hybrid Discrete Differential Evolution Approach for the Single Machine Total Stepwise Tardiness Problem with Release Dates**. In *Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC 2021)*, pages 652–659. IEEE, 2021. (34)

[88] G. SRIVASTAVA, A. SINGH, AND R. MALLIPEDDI. **NSGA-II with objective-specific variation operators for multiobjective vehicle routing problem with time windows**. *Expert Systems with Applications*, **176**:114779, 2021. (34)

[89] G. SRIVASTAVA, P. VENKATESH, AND A. SINGH. **An evolution strategy based approach for cover scheduling problem in wireless sensor networks**. *International Journal of Machine Learning and Cybernetics*, **11**(9):1981–2006, 2020. (34)

[90] V. PANDIRI, A. SINGH, AND A. ROSSI. **Two hybrid metaheuristic approaches for the covering salesman problem**. *Neural Computing and Applications*, **32**(19):15643–15663, 2020. (34, 62)

[91] A. ROSSI, A. SINGH, AND M. SEVAUX. **Focus distance-aware lifetime maximization of video camera-based wireless sensor networks**. *Journal of Heuristics*, **27**(1-2):5–30, 2021. (34, 62)

[92] S. N. CHAURASIA AND A. SINGH. **A hybrid evolutionary algorithm with guided mutation for minimum weight dominating set**. *Applied Intelligence*, **43**(3):512–529, 2015. (34)

[93] A. SINGH, A. ROSSI, AND M. SEVAUX. **Matheuristic approaches for Q-coverage problem versions in wireless sensor networks**. *Engineering Optimization*, **45**(5):609–626, 2013. (34)

[94] T. H. GRUBESIC, A. T. MURRAY, W. A. PRIDEMORE, L. P. TABB, Y. LIU, AND R. WEI. **Alcohol beverage distribution control, privatization and the geographic distribution of alcohol outlets**. *BMC Public Health*, **12**:1015, 2012. (36)

[95] TONY H GRUBESIC AND ALAN T MURRAY. **Sex offender residency and spatial equity**. *Applied Spatial Analysis and Policy*, **1**(3):175–192, 2008. (36)

[96] R. L. CHURCH AND J. L. COHON. **Multiobjective location analysis of regional energy facility siting problems**. Technical report, Brookhaven National Lab., Upton, NY (USA), 1976. (48)

[97] R. L. CHURCH AND R. S. GARFINKEL. **Locating an obnoxious facility on a network**. *Transportation Science*, **12**(2):107–118, 1978. (48)

[98] E. ERKUT. **The discrete p-dispersion problem**. *European Journal of Operational Research*, **46**(1):48–60, 1990. (48)

[99] J. R. CURRENT AND J. E. STORBECK. **A multiobjective approach to design franchise outlet networks**. *Journal of the Operational Research Society*, **45**(1):71–81, 1994. (48)

[100] D. JOSEPH, J. MEIDANIS, AND P. TIWARI. **Determining DNA sequence similarity using maximum independent set algorithms for interval graphs**. In *Scandinavian Workshop on Algorithm Theory*, pages 326–337. Springer, 1992. (48)

[101] F. BARAHONA, A. WEINTRAUB, AND R. EPSTEIN. **Habitat dispersion in forest planning and the stable set problem**. *Operations Research*, **40**(1-supplement-1):S14–S21, 1992. (48)

[102] Z. DREZNER. **The p-cover problem**. *European Journal of Operational Research*, **26**(2):312 – 313, 1986. (60)

[103] R. L. CHURCH AND R. S. GARFINKEL. **Locating an obnoxious facility on a network**. *Transportation science*, **12**(2):107–118, 1978. (60, 61)

[104] E. ERKUT AND S. NEUMAN. **Analytical models for locating undesirable facilities**. *European Journal of Operational Research*, **40**(3):275–291, 1989. (60, 61)

[105] P. HANSEN AND J. COHON. **On the location of an obnoxious facility**. *Sistemi urbani Napoli*, (3):299–317, 1981. (61)

[106] Z. DREZNER AND G. O. WESOLOWSKY. **Obnoxious facility location in the interior of a planar network**. *Journal of Regional Science*, **35**(4):675–688, 1995. (61)

[107] Z. DREZNER AND A. SUZUKI. **The big triangle small triangle method for the solution of nonconvex facility location problems**. *Operations Research*, **52**(1):128–135, 2004. (61)

[108] T. DREZNER, Z. DREZNER, AND C. H. SCOTT. **Location of a facility minimizing nuisance to or from a planar network**. *Computers & Operations Research*, **36**(1):135–148, 2009. (61)

[109] Z. DREZNER AND G. O. WESOLOWSKY. **The location of an obnoxious facility with rectangular distances**. *Journal of Regional Science*, **23**(2):241–248, 1983. (61)

[110] M. J. KAISER AND T. L. MORIN. **Locating an obnoxious facility**. *Applied mathematics letters*, **5**(3):25–26, 1992. (61)

[111] F. PLASTRIA. **Optimal location of undesirable facilities: a selective overview**. *JORBEL-Belgian Journal of Operations Research, Statistics, and Computer Science*, **36**(2-3):109–127, 1996. (61)

[112] E. CARRIZOSA AND F. PLASTRIA. **Locating an undesirable facility by generalized cutting planes**. *Mathematics of operations research*, **23**(3):680–694, 1998. (61)

[113] M. I. SHAMOS AND D. HOEY. **Closest-point problems**. In *16th Annual Symposium on Foundations of Computer Science (SFCS 1975)*, pages 151–162. IEEE, 1975. (61)

[114] Z. DREZNER, C. H. SCOTT, AND J. TURNER. **Mixed planar and network single-facility location problems**. *Networks*, **68**(4):271–282, 2016. (61)

[115] J. M. COLMENAR, P. GREISTORFER, R. MARTÍ, AND A. DUARTE. **Advanced greedy randomized adaptive search procedure for the obnoxious p-median problem**. *European Journal of Operational Research*, **252**(2):432–442, 2016. (61)

[116] T. DREZNER, Z. DREZNER, AND A. SCHÖBEL. **The Weber obnoxious facility location model: A big arc small arc approach**. *Computers & Operations Research*, **98**:240–250, 2018. (61)

[117] A. WEBER AND C.J. FRIEDRICH. *Alfred Weber's Theory of the Location of Industries*. Materials for the study of business. University of Chicago Press, 1929. (61)

[118] G. O. WESOLOWSKY. **The Weber Problem: History and Perspectives.** *Computers & Operations Research*, **1**(1):5–23, 1993. (61)

[119] R. L. CHURCH. **Understanding the Weber location paradigm**. In *Contributions to Location Analysis*, pages 69–88. Springer, 2019. (61)

[120] T. DREZNER, Z. DREZNER, AND P. KALCZYNSKI. **Multiple obnoxious facilities location: A cooperative model**. *IISE Transactions*, **52**(12):1403–1412, 2020. (62)

[121] S. N. CHAURASIA AND A. SINGH. **Hybrid evolutionary approaches for the single machine order acceptance and scheduling problem**. *Applied Soft Computing*, **52**:725–747, 2017. (62)

[122] A. SINGH, A. ROSSI, AND M. SEVAUX. **Matheuristic approaches for Q-coverage problem versions in wireless sensor networks**. *Engineering Optimization*, **45**(5):609–626, 2013. (62)

[123] K. SINGH AND S. SUNDAR. **A hybrid genetic algorithm for the degree-constrained minimum spanning tree problem**. *Soft Computing*, **24**(3):2169–2186, 2020. (62)

[124] K. SINGH AND S. SUNDAR. **A hybrid steady-state genetic algorithm for the min-degree constrained minimum spanning tree problem**. *European Journal of Operational Research*, **276**(1):88–105, 2019. (62)

[125] DAVID E. G. AND KALYANMOY D. **A comparative analysis of selection schemes used in genetic algorithms**. In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991. (68)

[126] H.Y. KIM. **Statistical notes for clinical researchers: Nonparametric statistical methods: 2. Nonparametric methods for comparing three or more groups and repeated measures**. *Restorative Dentistry and Endodontics*, **39**(4):329–332, 2014. (81)

[127] J. H. MCDONALD. *Handbook of biological statistics*. sparky house publishing Baltimore, MD, 2009. (81)

[128] S. L. HAKIMI. **Optimum locations of switching centers and the absolute centers and medians of a graph**. *Operations research*, **12**(3):450–459, 1964. (83)

[129] O. KARIV AND S. L. HAKIMI. **An algorithmic approach to network location problems. I: The p-centers**. *SIAM Journal on Applied Mathematics*, **37**(3):513–538, 1979. (83)

[130] C. S. REVELLE AND R. W. SWAIN. **Central facilities location**. *Geographical analysis*, **2**(1):30–42, 1970. (83)

[131] G. CORNUEJOLS, G. L. NEMHAUSER, AND L. A. WOLSEY. **A canonical representation of simple plant location problems and its applications**. *SIAM Journal on Algebraic Discrete Methods*, **1**(3):261–272, 1980. (83)

[132] ENRIQUE D. AND JOSÉ M. **A neural model for the p-median problem**. *Computers & Operations Research*, **35**(2):404–416, 2008. (83)

[133] S. ELLOUMI. **A tighter formulation of the p-median problem**. *Journal of combinatorial optimization*, **19**(1):69–83, 2010. (83)

[134] K. GHOSEIRI AND S. F. GHANNADPOUR. **An efficient heuristic method for capacitated P-Median problem**. *International Journal of Management Science and Engineering Management*, **4**(1):72–80, 2009. (83)

[135] K. FLESZAR AND K. S. HINDI. **An effective VNS for the capacitated p-median problem**. *European Journal of Operational Research*, **191**(3):612–622, 2008. (83)

[136] M.J. CANÓS, C. IVORRA, AND V. LIERN. **An exact algorithm for the fuzzy p-median problem**. *European Journal of Operational Research*, **116**(1):80–86, 1999. (83, 84)

[137] J.M. CADENAS, M.J. CANÓS, M.C. GARRIDO, C. IVORRA, AND V. LIERN. **Soft-computing based heuristics for location on networks: The p-median problem**. *Applied Soft Computing*, **11**(2):1540–1547, 2011. (84)

[138] J. R. WEAVER AND R. L. CHURCH. **A median location model with nonclosest facility service**. *Transportation Science*, **19**(1):58–74, 1985. (84)

[139] J. BRIMBERG, A. MAIER, AND A. SCHÖBEL. **When closest is not always the best: The distributed p-median problem**. *Journal of the Operational Research Society*, **72**(1):200–216, 2021. (84)

[140] R. L. CHURCH, M. P. SCAPARRA, AND R. S. MIDDLETON. **Identifying critical infrastructure: the median and covering facility interdiction problems**. *Annals of the Association of American Geographers*, **94**(3):491–502, 2004. (84, 86)

[141] L. V. SNYDER AND M. S. DASKIN. **Reliability models for facility location: the expected failure cost case**. *Transportation Science*, **39**(3):400–416, 2005. (84, 85, 86, 88, 114, 117)

[142] J. ALCARAZ, M. LANDETE, AND J. F. MONGE. **Design and analysis of hybrid metaheuristics for the reliability p-median problem**. *European Journal of Operational Research*, **222**(1):54–64, 2012. (85, 86, 88, 93, 97, 98, 99, 101, 124)

[143] C.J. COLBOURN. *The Combinatorics of Network Reliability*. Oxford University Press, New York, 1987. (85)

[144] D. R. SHIER. *Network reliability and algebraic structures*. Clarendon Press, 1991. (85)

[145] M. L. SHOOMAN. *Reliability of computer systems and networks: fault tolerance, analysis, and design*. John Wiley & Sons, 2003. (85)

[146] J. E. MURIEL-VILLEGAS, K. C. ALVAREZ-URIBE, C. E. PATIÑO-RODRÍGUEZ, AND J. G. VILLEGAS. **Analysis of transportation networks subject to natural hazards–Insights from a Colombian case**. *Reliability Engineering & System Safety*, **152**:151–165, 2016. (85)

[147] H. WAKABAYASHI AND Y. IIDA. **Upper and lower bounds of terminal reliability of road networks: an efficient method with Boolean algebra**. *Journal of natural disaster science*, **14**(1), 1992. (85)

[148] Z. P. DU AND A. NICHOLSON. **Degradable transportation systems: sensitivity and reliability analysis**. *Transportation Research Part B: Methodological*, **31**(3):225–237, 1997. (85)

[149] R. KONDO, Y. SHIOMI, AND N. UNO. **Network evaluation based on connectivity reliability and accessibility**. In *Network reliability in practice*, pages 131–149. Springer, 2012. (85)

[150] Y. S. QIAN, M. WANG, H. X. KANG, J. W. ZENG, AND Y. F. LIU. **Study on the road network connectivity reliability of valley city based on complex network**. *Mathematical Problems in Engineering*, **2012**, 2012. (85)

[151] H. A. EISELT, M. GENDREAU, AND G. LAPORTE. **Optimal location of facilities on a network with an unreliable node or link**. *Information processing letters*, **58**(2):71–74, 1996. (85)

[152] O. BERMAN, D. KRASS, AND M. B. C. MENEZES. **Facility reliability issues in network p-median problems: Strategic centralization and co-location effects**. *Operations research*, **55**(2):332–350, 2007. (85)

[153] L. V. SNYDER, M. S. DASKIN, AND C. P. TEO. **The stochastic location model with risk pooling**. *European Journal of Operational Research*, **179**(3):1221–1238, 2007. (85)

[154] P. PENG, L. V. SNYDER, A. LIM, AND Z. LIU. **Reliable logistics networks design with facility disruptions**. *Transportation Research Part B: Methodological*, **45**(8):1190–1211, 2011. (85)

[155] X. WANG AND Y. OUYANG. **A continuum approximation approach to competitive facility location design under facility disruption risks**. *Transportation Research Part B: Methodological*, **50**:90–103, 2013. (85)

[156] S. AN, N. CUI, Y. BAI, W. XIE, M. CHEN, AND Y. OUYANG. **Reliable emergency service facility location under facility disruption, en-route congestion and in-facility queuing**. *Transportation research part E: logistics and transportation review*, **82**:199–216, 2015. (85)

[157] Y. ZHANG, O. BERMAN, AND V. VERTER. **Incorporating congestion in preventive healthcare facility network design**. *European Journal of Operational Research*, **198**(3):922–935, 2009. (85)

[158] X. LI, Y. OUYANG, AND F. PENG. **A supporting station model for reliable infrastructure location design under interdependent disruptions**. *Procedia-Social and Behavioral Sciences*, **80**:25–40, 2013. (85)

[159] O. BERMAN, D. KRASS, AND M. B. C. MENEZES. **Location and reliability problems on a line: Impact of objectives and correlated failures on optimal location patterns**. *Omega*, **41**(4):766–779, 2013. (85)

[160] D. GADE AND E. A. POHL. **Sample average approximation applied to the capacitated-facilities location problem with unreliable facilities**. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, **223**(4):259–269, 2009. (85)

[161] N. AYDIN AND A. MURAT. **A swarm intelligence based sample average approximation algorithm for the capacitated reliable facility location problem**. *International Journal of Production Economics*, **145**(1):173–183, 2013. (85)

[162] M. LIM, M. S. DASKIN, A. BASSAMBOO, AND S. CHOPRA. **A facility reliability problem: Formulation, properties, and algorithm**. *Naval Research Logistics (NRL)*, **57**(1):58–70, 2010. (85)

[163] G. CHEN, M. S. DASKIN, Z. J. M. SHEN, AND S. URYASEV. **The α-reliable mean-excess regret model for stochastic facility location modeling**. *Naval Research Logistics (NRL)*, **53**(7):617–626, 2006. (85, 86)

[164] D. AKSEN AND N. ARAS. **A bilevel fixed charge location model for facilities under imminent attack**. *Computers & Operations Research*, **39**(7):1364–1381, 2012. (86)

[165] F. LIBERATORE, M. P. SCAPARRA, AND M. S. DASKIN. **Analysis of facility protection strategies against an uncertain number of attacks: The stochastic R-interdiction median problem with fortification**. *Computers & Operations Research*, **38**(1):357–366, 2011. (86)

[166] Q. LI, B. ZENG, AND A. SAVACHKIN. **Reliable facility location design under disruptions**. *Computers & Operations Research*, **40**(4):901–909, 2013. (86)

[167] R. C. LARSON. **A hypercube queuing model for facility location and redistricting in urban emergency services**. *Computers & Operations Research*, **1**(1):67–95, 1974. (86)

[168] R. C. LARSON. **Approximating the performance of urban emergency service systems**. *Operations research*, **23**(5):845–868, 1975. (86)

[169] O. BERMAN AND D. KRASS. **Facility location problems with stochastic demands and congestion**. *Facility location: applications and theory*, page 329, 2001. (86)

[170] BADR AFIFY, SUJOY RAY, ANDREI SOEANU, ANJALI AWASTHI, MOURAD DEBBABI, AND MOHAMAD ALLOUCHE. **Evolutionary learning algorithm for reliable facility location under disruption**. *Expert Systems with Applications*, **115**:223–244, 2019. (86)

[171] BADR AFIFY, ANDREI SOEANU, AND ANJALI AWASTHI. **Separation linearization approach for the capacitated facility location problem under disruption**. *Expert Systems with Applications*, **169**:114187, 2021. (86)

[172] W. DAI, G. R. XUE, Q. YANG, AND Y. YU. **Transferring naive bayes classifiers for text classification**. In *AAAI*, **7**, pages 540–545, 2007. (90)

[173] S. XU. **Bayesian Naive Bayes classifiers to text classification**. *Journal of Information Science*, **44**(1):48–59, 2018. (90)

[174] H. CHEN, S. HU, R. HUA, AND X. ZHAO. **Improved naive Bayes classification algorithm for traffic risk management**. *EURASIP Journal on Advances in Signal Processing*, **2021**(1):1–12, 2021. (90)

[175] W. WEI, S. VISWESWARAN, AND G. F. COOPER. **The application of naive Bayes model averaging to predict Alzheimer's disease from genome-wide data**. *Journal of the American Medical Informatics Association*, **18**(4):370–375, 2011. (90)

[176] X. XIE, J. W. K HO, C. MURPHY, G. KAISER, B. XU, AND T. Y. CHEN. **Testing and validating machine learning classifiers by metamorphic testing**. *Journal of Systems and Software*, **84**(4):544–558, 2011. (90)

[177] V. PANDIRI AND A. SINGH. **An artificial bee colony algorithm with variable degree of perturbation for the generalized covering traveling salesman problem**. *Applied Soft Computing*, **78**:481–495, 2019. (93)

[178] Y. KOCHETOV, T. LEVANOVA, E. ALEKSEEVA, AND M. LORESH. **Large neighborhood local search for the p-median problem**. *Yugoslav Journal of Operations Research*, **15**(1):53–63, 2005. (96, 112)

[179] J. E. BEASLEY. **A note on solving large p-median problems**. *European Journal of Operational Research*, **21**(2):270–273, 1985. (97)

[180] O. BERMAN, D. KRASS, AND M. B. C. MENEZES. **Locating facilities in the presence of disruptions and incomplete information**. *Decision Sciences*, **40**(4):845–868, 2009. (102, 103)

[181] M. ALBAREDA-SAMBOLA, Y. HINOJOSA, AND J. PUERTO. **The reliable p-median problem with at-facility service**. *European Journal of Operational Research*, **245**(3):656–666, 2015. (103, 104, 106, 107, 114, 115, 117, 118, 119)

[182] T. DAVIDOVIĆ, D. RAMLJAK, M. ŠELMIĆ, AND D. TEODOROVIĆ. **Bee colony optimization for the p-center problem**. *Computers & Operations Research*, **38**(10):1367–1376, 2011. (108)

[183] V. PANDIRI AND A. SINGH. **Two multi-start heuristics for the k-traveling salesman problem**. *OPSEARCH*, **57**(4):1164–1204, 2020. (113)

List of Publications

[1] **EDUKONDALU CHAPPIDI** AND ALOK SINGH. **Discrete differential evolution-based solution for anti-covering location problem.** *Proceedings of the 10th International Conference on Soft Computing for Problem Solving (SocProS 2020), Advances in Intelligent Systems and Computing,* ***1392****: 607-620, 2021, Springer.*

[2] **EDUKONDALU CHAPPIDI**, ALOK SINGH AND RAMMOHAN MALLIPEDDI. **Intelligent optimization algorithms for disruptive anti-covering location problem.** To appear in *Proceedings of the 19th International Conference on Distributed Computing and Intelligent Technology (ICDCIT 2023), Lecture Notes in Computer Science, 2023, Springer.*

[3] **EDUKONDALU CHAPPIDI** AND ALOK SINGH. **Evolutionary approaches for the weighted anti-covering location problem.** To appear in *Evolutionary Intelligence, Springer.*

[4] **EDUKONDALU CHAPPIDI** AND ALOK SINGH. **An evolutionary approach for obnoxious cooperative maximum covering location problem.** *Applied Intelligence,* ***52****: 16651–16666, 2022, Springer.*

[5] **EDUKONDALU CHAPPIDI** AND ALOK SINGH. **A hyper-heuristic based approach with naive Bayes classifier for the reliability p-median problem.** Communicated to *Applied Intelligence, Springer.*

[6] **EDUKONDALU CHAPPIDI** AND ALOK SINGH. **Two multi-start hyper-heuristic approaches for the reliable p-median problem with at-facility service.** Communicated to *Operational Research, Springer.*

Heuristics for Facility Location Problems

by Mr. Edukondalu Chappidi

Papers at S.No 1 to 4 report the work of the thesis. Excluding these papers similarity will be 7%.

Heuristics for Facility Location Problems

ASingh
21/12/2022
(Supervisor)

Dr. Alok Singh
Professor
School of Computer & Information Sciences
University of Hyderabad
Hyderabad-500 046, Telangana, India.

ORIGINALITY REPORT

32%	23%	32%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	link.springer.com Internet Source	11%
2	"Soft Computing for Problem Solving", Springer Science and Business Media LLC, 2021 Publication	7%
3	Edukondalu Chappidi, Alok Singh. "An evolutionary approach for obnoxious cooperative maximum covering location problem", Applied Intelligence, 2022 Publication	5%
4	Edukondalu Chappidi, Alok Singh. "Evolutionary approaches for the weighted anti-covering location problem", Evolutionary Intelligence, 2022 Publication	2%
5	Submitted to University of Hyderabad, Hyderabad Student Paper	1%
6	upcommons.upc.edu Internet Source	<1%

7	"Location Science", Springer Science and Business Media LLC, 2019 Publication	<1 %
8	eprints.nottingham.ac.uk Internet Source	<1 %
9	Lawrence V. Snyder, Mark S. Daskin. "Reliability Models for Facility Location: The Expected Failure Cost Case", Transportation Science, 2005 Publication	<1 %
10	escholarship.org Internet Source	<1 %
11	Javier Alcaraz, Mercedes Landete, Juan F. Monge. "Design and analysis of hybrid metaheuristics for the Reliability p-Median Problem", European Journal of Operational Research, 2012 Publication	<1 %
12	"Handbook of Heuristics", Springer Science and Business Media LLC, 2018 Publication	<1 %
13	Venkatesh Pandiri, Alok Singh. "Two multi-start heuristics for the k-traveling salesman problem", OPSEARCH, 2020 Publication	<1 %
14	dokumen.pub Internet Source	<1 %

15	etheses.whiterose.ac.uk Internet Source	<1 %
16	"Location Science", Springer Science and Business Media LLC, 2015 Publication	<1 %
17	"Simulated Evolution and Learning", Springer Science and Business Media LLC, 2014 Publication	<1 %
18	Sachchida Nand Chaurasia, Alok Singh. "A hybrid evolutionary approach to the registration area planning problem", Applied Intelligence, 2014 Publication	<1 %
19	Studies in Computational Intelligence, 2008. Publication	<1 %
20	Data Mining, 2015. Publication	<1 %
21	B. Jayalakshmi, Alok Singh. "Two swarm intelligence-based approaches for the *p*-centre problem", International Journal of Swarm Intelligence, 2018 Publication	<1 %
22	Lecture Notes in Computer Science, 2007. Publication	<1 %
23	Lecture Notes in Computer Science, 2011. Publication	<1 %

24	ssli.ee.washington.edu Internet Source	<1 %
25	Sumanta Basu, Megha Sharma, Partha Sarathi Ghosh. "Metaheuristic applications on discrete facility location problems: a survey", OPSEARCH, 2014 Publication	<1 %
26	Ting L. Lei, Daoqin Tong. "Hedging against service disruptions: an expected median location problem with site-dependent failure probabilities", Journal of Geographical Systems, 2012 Publication	<1 %
27	Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, Rong Qu. "Hyper-heuristics: a survey of the state of the art", Journal of the Operational Research Society, 2017 Publication	<1 %
28	Preeti Ravindranath Khorjuvenkar, Alok Singh. "A Hybrid Swarm Intelligence Approach for Anti-Covering Location Problem", 2019 Innovations in Power and Advanced Computing Technologies (i-PACT), 2019 Publication	<1 %
29	repositorio.unb.br Internet Source	<1 %

30	"Computer and Information Sciences", Springer Science and Business Media LLC, 2016 Publication	<1 %
31	"Foundations of Location Analysis", Springer Science and Business Media LLC, 2011 Publication	<1 %
32	Operations Research Proceedings, 2006. Publication	<1 %
33	Sachchida Nand Chaurasia, Shyam Sundar, Alok Singh. "Hybrid metaheuristic approaches for the single machine total stepwise tardiness problem with release dates", Operational Research, 2016 Publication	<1 %
34	Lawrence .V Snyder, Zuo - Jun Max Shen. "Fundamentals of Supply Chain Theory", Wiley, 2019 Publication	<1 %
35	C.S. ReVelle, H.A. Eiselt, M.S. Daskin. "A bibliography for some fundamental problem categories in discrete location science", European Journal of Operational Research, 2008 Publication	<1 %
36	Gaurav Srivastava, Alok Singh, Rammohan Mallipeddi. "A Hybrid Discrete Differential	<1 %

Evolution Approach for the Single Machine Total Stepwise Tardiness Problem with Release Dates", 2021 IEEE Congress on Evolutionary Computation (CEC), 2021
Publication

37	"Hybrid Metaheuristics", Springer Science and Business Media LLC, 2010 Publication	<1 %
38	Dasari Kasi Viswanath, Pandiri Venkatesh, Alok Singh. "Multi-Start Heuristics for the Profitable Tour Problem", Swarm and Evolutionary Computation, 2021 Publication	<1 %
39	hdl.handle.net Internet Source	<1 %
40	ALOK SINGH, ASHOK KUMAR GUPTA. "A HYBRID HEURISTIC FOR THE MINIMUM WEIGHT VERTEX COVER PROBLEM", Asia-Pacific Journal of Operational Research, 2011 Publication	<1 %
41	Alcaraz, J.. "Design and analysis of hybrid metaheuristics for the Reliability p-Median Problem", European Journal of Operational Research, 20121001 Publication	<1 %
42	Xueping Li, Kaike Zhang. "A sample average approximation approach for supply chain	<1 %

network design with facility disruptions", Computers & Industrial Engineering, 2018
Publication

43	Parallel Problem Solving from Nature – PPSN XI, 2010. Publication	<1 %
44	"Swarm, Evolutionary, and Memetic Computing", Springer Science and Business Media LLC, 2011 Publication	<1 %
45	Alok Singh, Ashok Kumar Gupta. "A hybrid heuristic for the maximum clique problem", Journal of Heuristics, 2006 Publication	<1 %
46	www.rhsupplies.org Internet Source	<1 %
47	"AI 2018: Advances in Artificial Intelligence", Springer Science and Business Media LLC, 2018 Publication	<1 %
48	"Fuzzy Logic Hybrid Extensions of Neural and Optimization Algorithms: Theory and Applications", Springer Science and Business Media LLC, 2021 Publication	<1 %
49	Venkatesh Pandiri, Alok Singh. "A simple hyper-heuristic approach for a variant of	<1 %

many-to-many hub location-routing problem", Journal of Heuristics, 2021
Publication

50 megplanning.gov.in
Internet Source <1%

51 Alok Singh, André Rossi, Marc Sevaux. " Matheuristic approaches for -coverage problem versions in wireless sensor networks ", Engineering Optimization, 2013
Publication <1%

52 John H. Drake, Ahmed Kheiri, Ender Özcan, Edmund K. Burke. "Recent Advances in Selection Hyper-heuristics", European Journal of Operational Research, 2019
Publication <1%

53 Li, Zichuan, and Paul Schonfeld. "Hybrid simulated annealing and genetic algorithm for optimizing arterial signal timings under oversaturated traffic conditions : HYBRID SA AND GA FOR SIGNAL TIMING OPTIMIZATION", Journal of Advanced Transportation, 2014.
Publication <1%

54 Lecture Notes in Computer Science, 2014.
Publication <1%

55 Murat Oğuz, Tolga Bektaş, Julia A. Bennell. "Multicommodity flows and Benders decomposition for restricted continuous <1%

location problems", European Journal of Operational Research, 2018
Publication

56	Shyam Sundar, Alok Singh. "A hybrid heuristic for the set covering problem", Operational Research, 2010 Publication	<1 %
57	Gert W. Wolf. "Solving location - allocation problems with professional optimization software", Transactions in GIS, 2022 Publication	<1 %
58	Lecture Notes in Computer Science, 2016. Publication	<1 %
59	Ricardo B. Damm, Mauricio G.C. Resende, Débora P. Ronconi. "A biased random key genetic algorithm for the field technician scheduling problem", Computers & Operations Research, 2016 Publication	<1 %
60	dspace.nwu.ac.za Internet Source	<1 %
61	epdf.pub Internet Source	<1 %
62	riunet.upv.es Internet Source	<1 %

63	"Advances in Natural Computation", Springer Science and Business Media LLC, 2006 Publication	<1 %
64	Branka Dimitrijevic, Milos Nikolic, Katarina Vukadinovic, Ivana Vukicevic. "Locating dangerous goods with constant and variable impact radius", Vojnotehnicki glasnik, 2016 Publication	<1 %
65	Damgacioglu, Haluk, Derya Dinler, Nur Evin Ozdemirel, and Cem Iyigun. "A genetic algorithm for the uncapacitated single allocation planar hub location problem", Computers & Operations Research, 2015. Publication	<1 %
66	Pandiri Venkatesh, Alok Singh. "A Hyper-Heuristic Based Artificial Bee Colony Algorithm for k -Interconnected Multi-Depot Multi-Traveling Salesman Problem", Information Sciences, 2018 Publication	<1 %
67	Shyam Sundar, Alok Singh. "Metaheuristic Approaches for the Blockmodel Problem", IEEE Systems Journal, 2015 Publication	<1 %
68	researchrepository.wvu.edu Internet Source	<1 %

69	"Operations Research and Health Care", Springer Science and Business Media LLC, 2004 Publication	<1 %
70	"Soft Computing: Theories and Applications", Springer Science and Business Media LLC, 2019 Publication	<1 %
71	Breunig, U., V. Schmid, R.F. Hartl, and T. Vidal. "A large neighbourhood based heuristic for two-echelon routing problems", Computers & Operations Research, 2016. Publication	<1 %
72	Claudio Contardo. " Decremental Clustering for the Solution of -Dispersion Problems to Proven Optimality ", INFORMS Journal on Optimization, 2020 Publication	<1 %
73	Paola Garrone, Sergio Mariotti, Francesca Sgobbi. "Technological Innovation in Telecommunications: An Empirical Analysis of Specialisation Paths", Economics of Innovation and New Technology, 2002 Publication	<1 %
74	Patricia Domínguez-Marín. "Heuristic Procedures for Solving the Discrete Ordered	<1 %

Median Problem", Annals of Operations Research, 04/2005
Publication

75	Taiser Samer Jasim, Esam Taha Yassen, Sudad H. Abed. "A new competitive travelling salesmen problem based on metaheuristics", AIP Publishing, 2022 Publication	<1 %
76	library.cuhk.edu.hk Internet Source	<1 %
77	"Evolutionary Computation in Combinatorial Optimization", Springer Science and Business Media LLC, 2009 Publication	<1 %
78	"Evolutionary Computation in Combinatorial Optimization", Springer Science and Business Media LLC, 2013 Publication	<1 %
79	"Hybrid Intelligent Systems", Springer Science and Business Media LLC, 2020 Publication	<1 %
80	Berman, O.. "The minimum weighted covering location problem with distance constraints", Computers and Operations Research, 200802 Publication	<1 %
81	C.N. Vijeyamurthy. "Literature review of covering problem in operations	<1 %

management", International Journal of Services Economics and Management, 2010
Publication

82	Javier Alcaraz, Mercedes Landete, Juan F. Monge, José L. Sainz-Pardo. "Strengthening the reliability fixed-charge location model using clique constraints", Computers & Operations Research, 2015 Publication	<1 %
83	Juan A. Díaz, Dolores E. Luna. "Primal and dual bounds for the vertex p-median problem with balance constraints", Annals of Operations Research, 2016 Publication	<1 %
84	Proceedings of the Institute of Industrial Engineers Asian Conference 2013, 2013. Publication	<1 %
85	"Contributions to Location Analysis", Springer Science and Business Media LLC, 2019 Publication	<1 %
86	"Handbook of Metaheuristics", Springer Science and Business Media LLC, 2010 Publication	<1 %
87	B. Jayalakshmi, Alok Singh. "A hybrid artificial bee colony algorithm for the p-median problem with positive/negative weights", OPSEARCH, 2016 Publication	<1 %

88	Harris, Matthew, Regina Berretta, Mario Inostroza-Ponta, and Pablo Moscato. "A memetic algorithm for the quadratic assignment problem with parallel local search", 2015 IEEE Congress on Evolutionary Computation (CEC), 2015. Publication	<1 %
89	Mario Diaz, Hao Wang, Flavio P. Calmon, Lalitha Sankar. "On the Robustness of Information-Theoretic Privacy Measures and Mechanisms", IEEE Transactions on Information Theory, 2020 Publication	<1 %
90	Nasser R. Sabar, Masri Ayob, Graham Kendall, Rong Qu. "Automatic Design of a Hyper-Heuristic Framework With Gene Expression Programming for Combinatorial Optimization Problems", IEEE Transactions on Evolutionary Computation, 2015 Publication	<1 %
91	Reza Zanjirani Farahani, Nasrin Asgari, Nooshin Heidari, Mahtab Hosseininia, Mark Goh. "Covering problems in facility location: A review", Computers & Industrial Engineering, 2012 Publication	<1 %
92	Zhilei Ren, He Jiang, Jifeng Xuan, Yan Hu, Zhongxuan Luo. "New Insights Into	<1 %

Diversification of Hyper-Heuristics", IEEE Transactions on Cybernetics, 2014
Publication

93	imentaraddod.com Internet Source	<1%
94	people.sc.fsu.edu Internet Source	<1%
95	researchbank.swinburne.edu.au Internet Source	<1%

Exclude quotes	On	Exclude matches	< 14 words
Exclude bibliography	On		