Swarm Intelligence Techniques for Facility Location/Assignment Problems

A thesis submitted during 2016 to the University of Hyderabad in partial fulfilment of the award of a Ph.D. degree in School of Computer & Information Sciences

by

B. Jayalakshmi

School of Computer & Information Sciences

University of Hyderabad
P.O. Central University, Gachibowli
Hyderabad – 500046
Telangana
India

October 2016

CERTIFICATE

This is to certify that the thesis entitled **"Swarm Intelligence Techniques for Facility Location/Assignment Problems"** submitted by **B. Jayalakshmi** bearing **Reg. No. 11MCPC02** in partial fulfilment of the requirements for the award of **Doctor of Philosophy** in **Computer Science** is a bonafide work carried out by her under my supervision and guidance.

The thesis 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.

(Dr. Alok Singh)
Supervisor
School of Computer and Information Sciences
University of Hyderabad
Hyderabad – 500046, India

Dean
School of Computer and Information Sciences
University of Hyderabad
Hyderabad – 500046, India

DECLARATION

I, **B. Jayalakshmi**, hereby declare that this thesis entitled **"Swarm Intelligence Techniques for Facility Location/Assignment Problems"** submitted by me under the guidance and supervision of **Dr. 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 Shodhganga/INFLIBNET.

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

Date : **Name: B. Jayalakshmi**

Signature of the Student:

Reg. No.: 11MCPC02

Signature of the Supervisor:

Abstract

This thesis is focussed on solving some facility location/assignment problems through one particular swarm intelligence technique called Artificial Bee Colony (ABC) algorithm. Another swarm intelligence technique called Invasive Weed Optimization (IWO) algorithm has also been used to solve one problem. Facility location problems deal with locating facilities over potential locations subject to some constraints in order to optimize (minimize or maximize) a given objective function like cost, distance, service time, waiting time, coverage, etc. The facility assignment problems, on the other hand, seeks an assignment of a given set of facilities like concentrators, network bandwidth, etc. to a given set of customers subject to some constraints, so that a given objective function is optimized. The objective function may involve cost, distance, service time, load balancing, etc. or a combination of these factors. From a practical point of view, there is a lot of scope for location science/location theory. There are many application domains such as locating public facilities, commercial facilities, industrial plants, ware-houses, garbage dump yards and more where facility location models can be used to provide an effective solution.

ABC algorithm is a relatively new population based metaheuristic technique based on intelligent foraging behavior of honey bee swarm, which has been applied successfully to solve numerous combinatorial optimization problems in diverse domains. IWO algorithm is another new metaheuristic technique inspired by the phenomenon of survival and colonization of weeds in nature. It is shown in the literature that a powerful optimization algorithm can be obtained by capturing the properties of the invasive weeds. Since the development of original ABC and IWO algorithms, several different variants of these algorithms have been proposed in the literature,

In this thesis, we have studied four $\mathcal{NP}$-hard facility location problems which are based on either fundamental facility location models or their simple extensions. We choose these models because they are the fundamental and most extensively studied models of facility location, and they are the basis of many advanced models such as those involving capacity constraints, routing decisions, etc. We have also considered three $\mathcal{NP}$-hard facility assignment problems in this thesis, viz. terminal assignment problem and two ring loading problems. These problems originate in the design of fixed telecommunication networks and routing in fibre optical ring networks. These facility assignment problems are also closely related to a number of classical combinatorial optimization problems in different fields such as knapsack problem, task assignment problem, problem of assigning cells to switches in mobile communication networks.

We have considered seven problems in this thesis. These seven problems are as follows: p-median problem, p-median problem with positive/negative weights, cooperative maximum coverage location problem, p-center problem, terminal assignment problem and weighted ring edge-loading problem and weighted ring arc-loading problem. Out of these seven problems, first four problems are facility location problems, whereas remaining three problems are facility assignment problems.

We have developed ABC algorithm based approaches for all the seven problems. An IWO algorithm based approach is also presented for p-center problem. For each problem, we have compared our proposed approach/approaches with the state-of-the-art approaches on the standard benchmark instances of that problem. Computational results show the effectiveness of our proposed approaches. Our approaches can be easily extended to solve several related facility location and assignment problems.

To my late father ***B.Kanakaiah***

Acknowledgements

Even before I begin, I would like to express my sincere thanks to my supervisor Dr. Alok Singh for all his dedicated efforts in guiding me, sharing his immense knowledge and for exhibiting exceptional levels of patience during the entire process of my research. I am also grateful to him for his time, support and constant encouragement. I am glad to have got an opportunity to work under his guidance. Today, if I could deliver the result of my research in the form of this thesis, it's because of his support, continuous feedback and huge motivation provided during my study.

I would also like to thank my doctoral review committee members Prof. K. N. Murthy and Prof. S. K. Udgata for their probing queries, feedback, and suggestions which helped me to enhance the quality of my research from various perspectives. I am also thankful to other faculty members of the school for their support and encouragement.

I would like to extend my gratitude to the Dean of the School Prof. Arun Agarwal for providing us with all the essential facilities at the school especially the computing infrastructure and the well-trained staff that helped me to complete my research with ease and comfort. Due credit to the University for creating a research oriented computer science school and a library containing a large collection of research books & journals. Without these facilities, my journey as a research scholar would not have been smooth.

Several researchers from around the globe helped me in various ways and I am indebted to all of them. I am thankful to Prof. Mehmet Sevkli for providing the Galvao test instances for the p-median problem. My sincere thanks to Dr. Jörg Kalcsics for supplying the test instances and their detailed solution values for the cooperative maximum coverage location problem. I am grateful to Dr. E. M. Bernardino, who provided me the test

instances for the terminal assignment problem. I am obliged to Dr. A. M. Bernardino for giving the test instances for two ring loading problems. My special thanks to Dr. Jafar Fathali for clarifying some of our doubts on the test instances of p-median problem with positive/negative weights.

With my head high, I thank my late father Mr. B. Kanakaiah, he had been my path builder, and it was his constant drive that helped me to achieve my dream of earning a doctorate.

It is my privilege to thank my mother Mrs. Kamala, who brought me into this world and has been my friend, guide, mentor and supported me all the time.

I especially thank my husband K. Srinivas for his love, patience, and unstinting support. Without him, my dream of earning a doctorate would remain a dream forever and would never have taken this real shape. He has taught me the true meaning of sacrifice, and compromise.

I can not thank my son Dakshith enough, who spent several tiring days away from me and helped me to focus on my goal. I heartily bless him and feel sorry for the time we spent apart.

I also thank all the other family members for their affection, care, and encouragement. I owe this thesis to them.

I thank my fellow labmates Mr. Sachchinda Nand Chaurasia, Mr. Abobakr Khalil Nasr Al-Shamiri, Mr. Venkatesh Pandiri and Mr. Gaurav Srivastava for all the exciting and fruitful discussions, and for all the fun and learning, we had together.

I am particularly thankful to University Grant Commission for providing the financial assistance. My research work was supported by Rajiv Gandhi National Fellowship funded by University Grant Commission.

Special and humble thanks to all my friends and all those who have helped me directly or indirectly in this journey. Life would not have been happy and fruitful without your presence. God bless you all.

B Jayalakshmi

Contents

List of Figures

List of Tables

LIST OF TABLES

Chapter 1

Introduction

Over the last few decades operations researchers are striving towards developing new models, which can be mapped closely to the reality. Facility location is considered as one such branch of Operations Research (OR), which has been attracting more and more researchers over the years. Determining the location of facilities is among the most difficult and crucial decision in a project. Location decisions are difficult, because once the facility locations are fixed, it requires a lot of effort to relocate. Facility location deals with locating facilities to provide service to the customer in order to maintain a trade-off between the cost and service. Facility location models are widely used in both public and private sectors for locating a wide variety of facilities, e.g. fire stations, police stations, schools, banking facilities, restaurants, medical services, swimming pools, vehicle service centers, shopping malls, hazardous and waste material disposal, etc.

Finding the exact number of facilities to be located and proper location for each of them is vital, because, locating too many or too few facilities or locating the facilities inappropriately will result in degraded performance and/or increased cost. In industry applications, if too many facilities are deployed, the cost incurred will increase beyond the desirable value. If few facilities are opened, customer service can be degraded leading to loss of customers' goodwill which in turn leads to loss of customers. Mislocation of facilities will results in poor performance even though the correct number of facilities are used. Poor location decisions will have the severe effect on the cost as well as the performance. Facility location is considered to be of even higher importance when applied to the emergency type facilities like fire stations, ambulances, police stations,

etc. If very few facilities are utilized, and/or they are not located properly, then they can influence the life and death decisions [1].

The characteristics of a facility location problem will depend on the nature of the objectives that are being considered and constraints being put on the location of facilities. The optimization criteria of the location problem will depend on the type of application where it is actually applied. In the case of private applications maximization of the profit and acquiring large market values from opponents are the prime goals. Facilities like a warehouse or manufacturing plant can be located to fulfil the precisely defined objective, for example, in locating a warehouse a reasonable requirement of the objective is the minimization of the capital cost and/or distribution cost. Whereas in public sector, cost minimization, the integrity of service and efficiency of the system are the main criteria. But, it is very difficult to precisely state the objectives in case of locating emergency service type facilities, such as fire stations, and ambulances. Generally, the emergency service type facilities focus on providing the service to the public. Since these objectives are difficult to measure, there exist two measures in the location theory to define the objectives precisely in the case of public facilities. The first measure is, total weighted distance/time for travel to reach the facilities and the second measure is, distance/time that the farthest user from a facility has to travel to reach that facility [2]. In general, the facility location problems deal with the minimization of operational and locational costs required for total coverage by the facilities or with maximizing the coverage given a specified number of available facilities [3].

Different variants of the facility location models with constraints have been studied and solved using different methods in the literature [4] [5] [6] [7] [8] [9]. Almost, all variants of the facility location problems fall into the category of $\mathcal{NP}$-hard problems, and hence, the scope of exact approaches are limited. Heuristics and metaheuristic techniques are used to solve most of the facility location problems [10] [11] [12] [13]. Heuristic techniques are more problem-specific, whereas metaheuristic techniques make use of problem-related information as components in its framework, which is independent of the problem. There has been a significant amount of research on metaheuristic based solution approaches to solve the facility location models in the literature [14] [13] [15] [16]. It has been shown in the literature that metaheuristic based approaches have performed well over the problem-specific heuristics. Review of different exact and heuristic methods applied for different variants of the problems is available in [17], [18],

[19]. Even though the location theory is considered as old, but, it is known fact that applications of these models are getting more attractive.

There is a wealth of literature available on facility location, location models and location analysis. Interested readers can refer to [1], [20], [17], [18], [21], [22], [23], [24] to find more about facility location models.

Closely related to facility location problems, there is another class of problems called facility assignment problems where a decision needs to be made about assigning a given set of service requests or clients to a given set of facilities subject to some constraints. Unlike the facility location problems, here the facility locations are either fixed or immaterial, and hence, location decision is not required and only assignment decision needs to be made. Assignment problems occur in diverse walks of life ranging from manufacturing, construction, transportation and logistics, internet applications, communication networks to code generation in compiler design. Though some of these problems can be mapped to assignment model or other models in linear programming [25], and hence, can be solved efficiently, most of these are $\mathcal{NP}$-hard. However, unlike the $\mathcal{NP}$-hard facility location problems which have been studied systematically as a group, each $\mathcal{NP}$-hard facility assignment problem is studied in isolation perhaps due to the diverse domains in which they are found.

Next two sections provide further details about these problems.

1.1 Facility location problems

Facility location problems consist of a finite set of users with service demands and a finite set of potential locations for the facilities that will provide service to the users by meeting the given constraints and achieving the desired objective. Usually, two decisions to be made in facility location problems, first is location decision, which determines where to locate the facilities and second is allocation or assignment decision, to find which customers are allocated to which facilities to satisfy their service demand. Thus, these models sometimes also referred to as location-allocation models [26].

Based on the number of candidate facilities, objective function, the solution space of the problem and many other decision factors, there are different types of models available. Depending on the solution space in which the problem is defined, there exists three models, viz. continuous models, discrete models, and network models.

In continuous facility location problems, a continuous space usually specified by the coordinates of a plane, is provided. Thus the objective of this model is to find points in continuous space for locating new facilities relative to other existing facilities located at given points in the continuous space. An overview of location theory in continuous space is available in [27]. A survey of solution methods for the continuous location-allocation problems can be found in [12].

A problem is known as a discrete facility location problem if a finite number of potential locations for facilities are provided as the solution space. These problems are concerned with choosing the best locations for facilities from a given set of potential locations subject to some constraints so as to optimize a given cost function. A survey of discrete facility location problems is available in [28]. Review of different variants of the discrete facility location problems along with their applications is presented in [17]. A survey on metaheuristic approaches for solving discrete facility location problems is available in [13].

The difference between continuous and discrete location models is that in the continuous case a function must be selected to measure the distance or cost between two points in the space, it can be Euclidean distance function or Rectilinear/Manhattan distance function, etc. Whereas in discrete model, original travel distance or cost between two points may be used. Continuous models are easy and fast in practice because they do not require any pre-computations to find the shortest paths, and also they are free from large databases of travel distances. But, these models are less precise than discrete ones, because of the approximation of travel distances by using a distance function [12].

In network problems, which are a generalization of discrete problems [29], facilities are located anywhere (on nodes and/or along the edges) on the network. The network can be a network of road transport, or a network of air transport, or a river transport, etc. [30]. Given a network, the network location problems deal with locating the facilities anywhere on the network (on nodes and/or along the edges) so as to optimize a given cost function while respecting the constraints imposed by the problem concerned.

In general, different types of objective functions are considered in location problems. Some of the commonly used objective functions are as follows: Minimizing the total set-up cost, minimizing the maximum distance between the facilities and demand points, minimizing the average distance travelled, minimizing the maximum distance travelled,

minimizing the number of located facilities, minimizing the service time, maximizing the service throughput, etc. Nowadays, there is an increasing interest in multi-objective network location problems also [30].

Covering problems are one of the most attractive models of the facility location. They are attractive because of their applicability to the real world problems specially for locating emergency type facilities. The coverage objective is introduced by Church and ReVelle [2]. In covering problems, customers receive services from facilities based on their distance. A customer is served by exactly one facility, not necessarily the nearest facility, within a given distance. This predefined distance is called as critical distance or coverage distance or coverage radius and this standard model is referred as the individual coverage model. In the case of certain types of facilities, a customer can be served by a group of facilities whose distance from customer falls under the predefined distance. In this case, facilities are considered to be cooperating to provide the service. These models are termed as cooperative coverage models [31]. The notion of coverage is related to a satisfying approach rather than finding a best possible method. There are many real life problems which can be formulated as covering models such as locating hospitals, police stations, schools, post offices, parks, libraries, banks, shopping malls, etc. [9]. More details, applications, and reviews about the covering problems are available in [32] [33] [34]. Details of cooperative coverage models are available in [35], [36], [31], and [37].

In covering problems, facilities need to be located as close as possible to customers they serve. However, there are some undesirable yet necessary facilities which need to be located as far as possible from the customers they serve. Locations of these facilities are determined using models known as obnoxious location models. Typical applications of such models include locations of nuclear reactors, waste/garbage material dumps, hazardous material disposal, sewage treatment plants, etc. A review of obnoxious facility location problems is given in [38].

Facility location problems are one of the most extensively studied combinatorial optimization problems. Different problem variants have been used to model a wide variety of real life problems. The treatment of the problem and the solution approaches for different variants are mostly based on their objective function and their constraints.

In this work, we have studied the three fundamental facility location models, viz. p-median problem, p-center problem and maximum coverage location problem. In these

models "distance" is considered as a measure in defining their objectives. However, this distance is considered differently in each of these models. The p-median problem uses "total distance" as a measure in defining its objective. This distance is defined as the sum or weighted sum of all distances between customers and their assigned facilities. The objective is to minimize either the sum of the distances between customers and their assigned facilities or a weighted sum of the distances between customers and their assigned facilities. Here, we have addressed both unweighted and weighted versions of p-median problem. On the other hand, p-center problem considers "maximum distance" as a measure, i.e., maximum distance over all distances between customers and their assigned facilities. Here the distance is usually unweighted, and it aims at minimizing the maximum distance over all the distances. The third model, maximum coverage location problem, considers "coverage distance" as a measure and a customer is considered as covered if, the distance at which he/she is located falls under the specified radius of the facility. It's objective is to locate a fixed number of facilities so as to maximize coverage (customer's covered) within a specified service distance. We choose these models because they are the most fundamental and broadly studied models of facility location, and, they are the basis for many advanced models such as those involving capacity constraints, routing decisions, etc. [13].

In locating emergency type services, the notion of coverage is well-established [2]. The proximity of the facility from a customer is used as a fundamental measure to decide whether a customer/demand point is covered or not. A demand point is considered to be covered if a facility is available within some specified distance/time standard to provide the service. Then the coverage is achieved if an individual from the demand point can receive the service from the facility within the time/distance standard or if the server at the facility can provide/reach the demand point within the specified time/distance standard. Applications of maximal covering location problem (MCLP) are discussed in [39].

In general, one should always focus on the shortest distance from the demand point to reach the facility, while using distance as the standard. It is important to consider that the distance that needs to be travelled from the farthest demand point to reach the facility should not be too long. Hence the concept of maximum distance/time is most applicable to the location of emergency type services as well as to the location of public services, such as post offices, schools, banks, libraries, etc., which can be modelled using

the p-center problem. For applications of p-center problem one can refer to [30] and [40].

There are many cases where we are interested in the average distance that a customer has to travel to receive the service or the average distance that a provider must travel to reach his/her clients. To address these kinds of problems, we turn to the p-median problem as minimizing the average distance is equivalent to minimizing the total distance. Applications of p-median problem is available in [30]. A comprehensive literature survey on metaheuristic approaches for p-median problem is available in [15].

Though we make the assignment decision in facility location problems also, facility assignment problems are different. In facility location problems, the objective function is influenced by both the location decision and assignment decision and these two decisions are inter-dependent. Facility assignment problems deal only with assignments and involve no location decision. The next section provides an overview of facility assignment problems.

1.2 Facility assignment problems

Facility assignment problems seek the assignment of a given set of clients to a given set of facilities under a given set of constraints so as to optimize a given cost function. In these problems, locations of facilities are either fixed or immaterial. Clients can be physical or non-physical and we have solved both kinds of facility assignment problems in this thesis. In general, the cost function is related to the overall performance of the system.

Some facility assignment problems can be mapped to classical assignment model or other models in linear programming and can be solved efficiently. The classical assignment problem aims at optimizing the cost of assigning a given number of jobs to an equal number of persons so that each job is assigned to exactly one person. Finding the best person for the job is the prime goal of the assignment model. For example, consider a set of workers having different levels of skill to perform various jobs. The assignment of a job that matches worker's skill perfectly requires less cost than the assignment of a job that does not match worker's skill. The objective of the model in this situation is to determine the minimum cost of assignment of jobs to workers.

The assignment model is actually a special case of the transportation model, in which the workers represent the sources, and the jobs represent the destinations. The supply of each source and the demand at each destination are exactly equal to 1. This model can be solved directly as a regular transportation model, but, the fact that all the supplies and demands are equal to 1 has led to the development of a simple solution algorithm called the *Hungarian method* [41]. The assignment model can be extended to situations where the number of jobs is not equal to the number of persons by introducing the suitable number of dummy persons or dummy jobs so that the number of jobs becomes equal to the number of persons. If the number of jobs is less than the number of persons then we introduce dummy jobs and if the number of persons is less than the number of jobs then we introduce dummy persons. The various costs associated with these dummy persons/jobs are chosen to be greater than or equal to the maximum cost associated with any pair of job and person. If the number of jobs is less than the number of persons then the persons who got dummy jobs in final solution will remain idle, and, if the number of persons is less than the number of jobs then the jobs assigned to dummy persons will remain unallocated.

There is a generalization of the assignment problem known as Generalized Assignment Problem (GAP). There are a number of jobs and a number of machines. Each machine has a capacity. A job can be assigned to any machine, and this assignment will incur some cost and fetch some profit. The cost and profit vary according to the assignment, i.e., the cost incurred and profit fetched depends on the machine to which a job is assigned. Moreover, the sum of the costs of the jobs assigned to a machine cannot exceed its capacity. The goal of GAP is to find an assignment of jobs to machines which maximizes the total profit without exceeding the capacity of even a single machine. Several facility assignment problems can be modelled using GAP or one of its variants. Likewise, some facility location problems can also be modelled as GAP [42]. However, GAP is $\mathcal{NP}$-hard, and, even determining whether there exists a feasible solution to GAP is $\mathcal{NP}$-complete [43].

In this work, we have addressed three facility assignment problems, one is terminal assignment problem, and the other two are ring loading problems, viz. weighted ring arc loading problem and weighted ring edge loading problem. The first problem is an example of a facility assignment problem with physical clients, viz. terminals, whereas

the other two involves non-physical clients, viz. communication demands. These problems originate in the design of fixed telecommunication networks and routing in fibre optical ring networks. These facility assignment problems are also closely related to a number of classical combinatorial optimization problems in different fields such as bin packing problem, the problem of assigning cells to switches in mobile communication networks.

Terminal assignment problem plays an important role in the design of telecommunication networks. In large centralized computer networks, a central computer serves numerous terminals or workstations. In such cases, concentrators are used to increase the efficiency of the network. Instead of connecting the terminals directly to the central computers, the terminals are connected to the concentrators and concentrators are connected to the central computer. The terminals and concentrators have fixed and known locations. The capacity requirements of each terminal are known. This requirement may vary from one terminal to another terminal. The maximum capacity of each concentrator and the costs of connecting each terminal to different concentrators are also known. The objective of the TA problem is to connect a given set of N terminals to a given set of M concentrators in such a way that the total cost of the network thus formed is minimum according to a given objective function. Many different approaches have been proposed in the literature to solve the TA problem.

Ring loading problems have their applications in data communication in fibre optical network with ring topologies. The first ring loading problem pertains to Synchronous Optical NETworking (SONET) and Synchronous Digital Hierarchy (SDH) based optical networks. Basically, SONET/SDH enforce a ring-based topology where nodes are connected via a ring of optical fibre cables. SONET/SDH rings are bi-directional, i.e., a message can be transmitted in either direction (clockwise or counter-clockwise) over the ring and the messages routed through clockwise direction compete with those messages routed through counter-clockwise direction for the common bandwidth. The amount of data transmitting through an edge in either direction at a particular instant is called its load at that instant. Obviously, the load on any edge can not exceed the available bandwidth. Weighted Ring Edge-Loading Problem (WRELP) is an important problem in this context that seeks to minimize the maximum load on any edge. Given a set of communication demands between various pairs of nodes, the WRELP consists in

routing the demands on the ring in either clockwise or counter-clockwise direction so that the maximum load over all the edges is minimized.

However, unlike SONET/SDH where there is a single bi-directional ring, a Resilient Packet Ring (RPR) based network consists of two distinct uni-directional rings (one clockwise and another counter-clockwise), each with its own bandwidth and the messages sent through clockwise ring do not compete with those messages sent through counter-clockwise ring for the common bandwidth. So bi-directionality in RPR is achieved by making use of these two rings while sending the messages. Clearly, in RPR we have to deal with directed edges or arcs, where an arc, depending on its direction, belongs to a clockwise or counter-clockwise ring. The amount of data transmitting through an arc at a particular instant is called its load at that instant. Weighted Ring Arc-Loading Problem (WRALP) in RPR is equivalent of WRELP in SONET/SDH. Given a set of communication demands between various pairs of nodes, the WRALP problem consists in routing the demands either through the clockwise ring or counter-clockwise ring so that the maximum load over all the arcs of both the rings is minimized.

This thesis presents swarm intelligence based approaches for various facility location/assignment problems considered. The next section provides a general introduction to swarm intelligence.

1.3 Swarm intelligence

Over the last two decades, developing metaheuristic techniques based on swarm intelligence has become a highly popular area of research. These techniques are inspired by the collective intelligence and self-organised behavior shown by a group of individuals or agents. Such a group of agents is referred to as a swarm. These agents can be living organisms (e.g., birds, insects, weeds, bacteria, etc.) or things (e.g., particles, water drops, etc.). These agents interact with one another and with the surroundings by exhibiting a stochastic behavior. Without any access to global information, these agents perform tasks parallelly (tasks are performed simultaneously by all the agents) and in a distributed manner (without any centralized control) by following simple rules which use only the local information exchanged among the individual agents. Interactions among such self-organized agents fuel the evolution of an intelligent global behavior without any knowledge to the individual agents. This form of collective intelligence is

referred to as *swarm intelligence*. The term swarm intelligence was introduced by Beni and Wang [44] in the context of cellular robotic systems. Division of labour and self-organization are two important properties of the swarm. Division of labour facilitates specialized agents of the swarm to perform a number of different tasks simultaneously. Division of labour can be clearly seen in social insects such as ants, bees, termites etc. Self-organization property of the swarm refers to a coherent global behavior evolved as a result of local interactions among the agents. Bonabeau et al. [45] listed four conditions for a swarm to exhibit self-organized behavior.

1. *Positive feedback* aids in the formation of purposeful structures comprising of many individual agents. The common examples of positive feedback include recruitment through dances in bees and pheromone trail reinforcement in ants.

2. *Negative feedback* helps avoid saturation. It offsets positive feedback and makes the collective pattern more stable. Food source exhaustion in case of foraging bees and pheromone trail evaporation in ants are common examples of negative feedback.

3. *Fluctuations* are random variations in behavior of agents in a swarm that helps in exploring new areas.

4. There should exist an *opportunity for multiple interactions* in a swarm. Normally, in a single interaction, information is exchanged only between two or at most few individuals. Therefore, in order to spread the information globally, multiple interactions are necessary.

This thesis is particularly concerned with one swarm intelligence technique, viz. Artificial Bee Colony (ABC) algorithm. Another swarm intelligence technique called Invasive Weed Optimization (IWO) algorithm is also used to solve one problem. The next two sections provide the overview of these two techniques.

1.3.1 Overview of artificial bee colony algorithm

One of the new population-based metaheuristic techniques inspired from the intelligent foraging behavior of honey bee swarm is artificial bee colony algorithm. In general, the foraging bees are classified into three categories, viz. employed, onlookers, and scouts.

Employed bees are the ones that exploit the food sources. They carry nectar from the food sources to the hive and share the information about their food sources with the onlooker bees. Onlooker bees are those bees who stay in the hive and wait for the employed bees to pass on the information about their food sources. The information about food sources is shared by employed bees through dancing in an area of the hive reserved specifically for this purpose. The duration and nature of the dance performed by an employed bee depend on the nectar content and position of the food source currently being exploited by that employed bee. Onlooker bees closely observe various dances performed in order to choose a food source. The onlookers tend to choose a food source with a probability proportional to the nectar content of that food source. So, if a food source has rich nectar content, it will attract more onlookers in comparison to a food source having poor nectar content. Scout bees are those bees which carry out the search for new food sources in the vicinity of the hive. Whenever a bee finds a food source irrespective of whether its a scout or onlooker, it starts exploiting it and becomes employed. Once a food source is completely exhausted, it is abandoned by all the employed bees associated with it. Such employed bees can become scouts or onlookers. Therefore, we can say that scout bees are responsible for exploration, whereas employed and onlooker bees are responsible for exploitation.

Karaboga [46] introduced the ABC algorithm in 2005 after deriving inspiration from the above described foraging behavior of honey bee swarm. Now, numerous extensions of this algorithm exist in literature, e.g. see [47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59] etc. Like real bees, in ABC algorithm, artificial bees are also categorized into scout, employed and onlooker bees with similar functions. In ABC algorithm, the food sources represent the possible solutions to the problem under consideration, and the quality of a food source represents the fitness of the represented solution. The employed bees are associated with food sources. There is always, a one-to-one association between food sources and employed bees, which means, the number of food sources is equal to the number of employed bees. Usually, but not always, the number of onlooker bees is also taken to be equal to the number of employed bees. The employed bee associated with an exhausted food source always becomes a scout and never an onlooker, and, as soon as this scout finds a new food source, it again becomes employed. The action carried out by a scout bee is modelled by generating a new food source and associating the scout bee in consideration with this newly generated food source to make it employed

again. We have used food source and solution interchangeably throughout this thesis. The ABC algorithm follows an iterative search process, which starts with initializing the employed bees with randomly generated food sources (solutions), then it repeats through the cycles of the employed bee and onlooker bee phases.

In the employed bee phase, each employed bee determines a food source in the proximity of its associated food source and evaluates its quality. The method of determining a new food source in the proximity of a particular food source depends on the problem under consideration. If the quality of the new food source is better than the current one, then the employed bee moves to the new food source leaving the old one. Otherwise, it remains at the old food source. When all the employed bees finish this process, then employed bee phase ends and onlooker bee phase begins.

In onlooker bee phase, onlookers select the food sources according to a probabilistic selection policy. The selection policy used by onlookers is designed in such a manner that good food sources will have more chance of being selected. As a result of such a probabilistic selection, good quality food sources will get more chance for selection by the onlookers. After all onlookers select the food sources, they determine the food sources in the proximity of their selected food sources and evaluate the fitness. Usually, the manner in which onlookers determine food sources in the proximity of their selected food sources is same as the employed bees, but it can be different also. Among all the neighboring food sources determined by the onlookers who chose food source i, and, the food source i itself, the best food source is determined. This best food source will be made the food source i for the next iteration. The onlooker bee phase ends when all the food sources are updated, and then the next iteration of the ABC algorithm begins.

The algorithm repeats itself through the cycles of the above mentioned two phases until the termination condition is satisfied. If a solution associated with any employed bee does not improve over some specific number of iterations, then that food source is considered as exhausted and it is discarded by its associated employed bee and that employed bee becomes the scout. Such a scout is converted back into employed bee by associating it with a new solution generated specifically for this purpose. Usually, this new solution is generated randomly in the same manner as an initial employed bee solution, but it can also be generated by perturbing either the current solution or the best solution.

In the employed bee phase, every solution is given a fair chance to improve itself, whereas in the onlooker bee phase, because of the biased selection policy used by the onlookers as mentioned above, good quality solutions get more chance to improve themselves in comparison to poor quality solutions as more number of onlookers gets attracted towards good quality solutions. This bias towards good quality solutions may produce better quality solutions faster, as there will be higher chances of finding even better solutions within the proximity of good solutions. However, if a solution is locally optimal, then no better solution exists in its proximity and any attempt to improve it will always fail. The concept of scout bees helps in getting rid of this situation. Instead of determining whether a solution is locally optimal or not with respect to the whole neighborhood which can be a computationally expensive process, a solution is deemed to be locally optimal if has not improved over a certain number of iterations. This solution is discarded by making its associated employed bee a scout. A new solution is generated for this scout bee to make it employed again. This new solution is generated either in the same manner as an initial solution or by perturbing an existing solution. Hence, the solutions which has not improved since long and which can be locally optimal are discarded utilizing the concept of scout bees. For a search process to be robust, the balance between the exploration and exploitation must be maintained. To maintain a proper balance between exploration and exploitation in the ABC algorithm, the number of iterations without improvement in the quality of a solution after which its associated employed bee leaves it and becomes a scout needs to be set appropriately.

The pseudo-code of ABC algorithm is presented in Algorithm 1, where n_e and n_o are the number of employed bees and number of onlooker bees respectively. The function *Determine_Neighbor_Solution(X)* returns a solution in the neighborhood of the solution X . The exact implementation of this function depends on the problem under consideration. The function *Scout()* generates a solution as per the policy used for generating scout bee solutions. Another function *select_index*(X_1, X_2,...,X_{n_e}) selects a solution from all the employed bee solutions X_1, X_2,...,X_{n_e} for an onlooker and returns the index of the solution selected. However, the exact implementation of this function depends on the selection policy used for selecting a food source for an onlooker bee. In all the ABC algorithm based approaches presented in this thesis, we have used *binary tournament selection method* for selecting a food source for an onlooker bee. The reason for using binary tournament selection method in place of roulette wheel

Algorithm 1: Pseudo-code of ABC algorithm

```
Generate n_e initial solutions e_1, e_2, ..., e_{n_e} randomly;
best_sol ← best solution among e_1, e_2, ..., e_{n_e} solutions;
while (termination criteria is not satisfied) do
    for (i ← 1 to n_e) do
        e' ← Determine_Neighbor_Solution(e_i);
        if (e' is better than e_i) then
            e_i ← e'
        else if (e_i has not changed over last limit iterations) then
            e_i ← Scout();
        if (e_i is better than best_sol) then
            best_sol ← e_i
    for (i ← 1 to n_o) do
        p_i ← select_index(e_1, e_2, ..., e_{n_e});
        On_i ← Determine_Neighbor_Solution(e_{p_i});
        if (On_i is better than best_sol) then
            best_sol ← On_i
    for (i ← 1 to n_o) do
        if (On_i is better than e_{p_i}) then
            e_{p_i} ← On_i
return best_sol;
```

selection method which is used commonly in ABC algorithm is that binary tournament selection method performs better in general and at the same time computationally less expensive [60].

A comprehensive survey of the ABC algorithm and its applications is available in [61].

1.3.2 Overview of invasive weed optimization algorithm

Invasive weed optimization (IWO) algorithm is another recently proposed metaheuristic technique. IWO algorithm is developed by Mehrabian and Lucas [62] in 2006. This technique is inspired by a phenomenon commonly observed in agriculture, i.e., colonization of invasive weeds. It is shown in the literature that a powerful optimization

algorithm can be obtained by capturing the properties of the invasive weeds. A weed in layman terminology is any unwanted plant growing in a place where it is not desired. But the use of the term weed is restricted in agriculture to only those plants which present a serious risk to survival and growth of desirable and cultivated plants with their intense and invasive manner of growth. Weeds are very robust and adaptive to the environment changes [63]. Weeds compete for resources like light, water, and nutrients with cultivated plants, thereby, adversely affecting the growth of cultivated plants. They grow before other desirable plants and will tap all critical resources and rapidly multiply their population, thereby, severely impacting the growth of cultivated plants in a negative manner. Weeds have the ability to produce numerous, long-lived, and easily transportable seeds. Weeds can produce the viable seeds even under adverse conditions and these seeds can survive long term. The seeds have the ability to survive even if parent plant is destroyed. Because of these properties, weeds find a place among the most robust plants in nature.

Motivated by this colonizing behavior of weeds, Mehrabian and Lucas [62] proposed IWO algorithm in 2006. Since then several different variants of the original IWO algorithm have been proposed in the literature, e.g. see [64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74] etc. In the IWO algorithm, each weed corresponds to a possible solution to the problem under consideration and the fitness of a weed is determined by the fitness of the solution it represents. IWO algorithm follows an iterative process which begins by generating a population of weed solutions randomly. Then an iterative process ensues where during each iteration, each weed produces a certain number of seeds (new solutions) according to its fitness. Usually, the number of seeds a weed can produce are mapped linearly from s_{min} to s_{max} according to its fitness. The seeds are produced in such a manner so that they are distributed in the search space according to a normally distribution with mean equal to the location of the producing weed and varying standard deviations. Following expression is used to determine the standard deviation (S.D.) σ at each iteration.

$$\sigma_{iter} := \left(\frac{iter_{max}-iter}{iter_{max}}\right)^{nmi} (\sigma_i - \sigma_f) + \sigma_f$$

Here $iter_{max}$ is maximum number of iterations, σ_{iter} is standard deviation at present iteration, and nmi is nonlinear modulation index, σ_i is the pre-defined initial value, σ_f is the pre-defined final value. Please note that the value of the standard deviation

Algorithm 2: Pseudo-code of IWO Approach

```
Generate n_i initial random weed solutions W_1, W_2, ..., W_{n_i};
best_sol := best among W_1, W_2, ..., W_{n_i} weed solutions;
n_c := n_i;
sort_weeds(n_c);
if n_c > n_max then
    n_c := n_max
while Termination condition is not reached do
    n_colony = n_c;
    for i := 1 to n_colony do
        seeds := no_of_seeds(W_i);
        for j := 1 to seeds do
            n_c := n_c + 1;
            W_{n_c} := generate_seed_solution(W_i);
            if W_{n_c} is better than best_sol then
                best_sol := W_{n_c}
    sort_weeds(n_c);
    if (n_c > n_max) then
        n_c := n_max
return best_sol;
```

is decreased from σ_i to σ_f over iterations of the algorithm so that the search space is explored thoroughly during initial iterations, and, it is reduced gradually over the iterations in order to focus the search in the neighborhood of good solutions only during final iterations. The exact manner in which seeds are produced depends on the problem under consideration. Once all weeds have finished producing their designated number of seeds, the newly produced seeds compete with weeds already existing in the colony for survival through a process known as competitive exclusion. If the number of weeds in the colony is less than the maximum number of weeds allowed in the colony n_{max}, then all the newly produced seeds are included in the colony of weeds as new weeds. However, once the number of weeds in the colony reaches n_{max}, only the fittest n_{max} weeds, among the existing ones and newly produced ones, are retained in the colony by the process of competitive exclusion. After this, another iteration of IWO algorithm begins. This process is repeated as long as termination condition is not satisfied.

The pseudo-code of IWO algorithm is given in Algorithm 2, where n_c represents the total number of weeds in the colony at any instance of time, and n_i is the number of weeds present in the colony at the time of initialization. n_{max} is the maximum number of weeds permissible in the colony. The function *sort_weeds(n_c)* sorts weeds according to non-increasing order of their fitness. Another function *no_of_seeds(W_i)* returns the number of seeds a weed W_i can produce, whereas *generate_seed_solution(W)* function generates a seed solution around the original solution W. The exact methods for determining the number of seeds that a weed can produce and for generating a seed solution vary according to the problem under consideration.

1.4 Scope of the thesis

This thesis is focussed on solving some $\mathcal{NP}$-hard facility location/assignment problems through one particular swarm intelligence technique, viz. artificial bee colony (ABC) algorithm. Invasive weed optimization (IWO) algorithm, which is another swarm intelligence technique, has also been used to solve one problem. We have considered seven problems in this thesis. Out of these seven problems, first four problems are facility location problems and the remaining three problems are facility assignment problems. These problems have many practical applications in diverse fields such as communication network design, transportation, logistic management, supply chain management, locating public/private facilities, cluster analysis, hazardous material disposal, waste material disposal, etc. We found through a literature review of these problems that either swarm intelligence based approaches particularly ABC algorithm do not exist, or existing techniques are not designed properly, and, hence, there is a scope for developing new approaches based on ABC algorithm which can outperform these existing approaches. This has motivated us to develop ABC algorithm based approaches for the problems considered in this thesis. We have developed either altogether new approaches or extended the existing approaches for these problems by incorporating appropriate changes in different components of ABC algorithm. All our approaches except one make use of local search heuristics in one way or the other. In addition, we have also incorporated problem-specific knowledge wherever appropriate in our approaches.

This thesis is divided into eight chapters beginning with this introductory chapter. Chapter 2 to Chapter 5 deal with the facility location problems, whereas Chapter 6 and

Chapter 7 address facility assignment problems. Chapter 8 is the concluding chapter. In the following, we outline the content of each of these chapters:

Chapter 2 deals with the p-median problem which is a well-known facility location problem. Given an undirected graph $G = (V, E)$ with $|V| = n$, the p-median problem seeks on this graph a set $Z \subseteq V$ of p vertices that minimizes the sum of distances from all the vertices in V to their respective closest vertices in Z. The vertices of the graph and the vertices in Z can be regarded as the demand points and the potential locations of facilities respectively, the objective is to select the locations of p facilities to serve n demand points, so that the sum of the distances of demand points from their nearest facilities is minimized. In this chapter, we have proposed an artificial bee colony algorithm based approach for solving this $\mathcal{NP}$-hard problem. We have tested the proposed algorithm on the OR-Library and Galvao p-median benchmark problems and the results are compared with those obtained with some other approaches available in the literature. The computational results show that the proposed algorithm outperforms other methods.

Chapter 3 addresses the p-median problem with positive/negative weights. In this chapter, we have considered the weighted case of p-median problem where every vertex in G has either a positive or a negative weight under two different objective functions, viz. minimizing the sum of the minimum weighted distances and minimizing the sum of the weighted minimum distances. A location problem with positive and negative weights on the vertices is useful in applications, where some facilities are non-attractive to some clients (facilities are obnoxious). We have extended the artificial bee colony algorithm developed in the previous chapter to the positive/negative weighted p-median problem and compared its performance with state-of-the-art approaches on the standard benchmark instances. Computational results show the effectiveness of our approach.

Chapter 4 proposes a hybrid artificial bee colony algorithm for the cooperative maximum covering location problem (CMCLP) on a network. In cooperative location covering problems, it is assumed that each facility generates a signal whose strength decreases with the increase in distance and a demand point is considered to be covered if the total signal strength received by it from various facilities exceeds a certain threshold. The objective of the CMCLP is to locate the facilities in such a way that maximizes

the total demands covered. The proposed hybrid approach obtained better quality solutions in comparison to two heuristic methods available in the literature.

Chapter 5 addresses the p-center problem. It is an important facility location problem which focusses on the quality of service. In this problem, the objective is to find a set of p centers on a weighted undirected graph $G = (V, E)$ in such a way that the maximum distance over all distances from vertices to their closest centers is minimized. In this chapter, we have proposed two metaheuristic approaches for the p-center problem. The first approach is based on artificial bee colony algorithm, whereas the latter approach is based on invasive weed optimization algorithm. The experiments performed on well-known benchmark problems show that our approaches are competitive with the state-of-the-art approaches available in the literature in terms of solution quality.

Chapter 6 is concerned with the solution of terminal assignment (TA) problem through an artificial bee colony algorithm based approach. TA problem is an important problem in the design of telecommunication networks. The problem consists in determining the best links for connecting a given set of terminals to a given set of concentrators subject to some constraints. Each terminal is assumed to have a positive weight, and each concentrator is assumed to have a fixed capacity. TA problem assigns terminals to concentrators in such a manner so that each terminal is assigned to exactly one concentrator and the sum of weights of terminals assigned to each concentrator should not be greater than its capacity. The goal of the TA problem is to minimize a weighted sum of distances of terminals from their assigned concentrators and the unbalance in loads on various concentrators as measured through a balance function. In comparison with the best methods available in the literature, the proposed approach obtained better quality solutions in shorter time.

In Chapter 7, we have proposed hybrid artificial bee colony algorithm based approaches for two $\mathcal{NP}$-hard problems arising in optical ring networks. These two problems fall under the category of ring loading problems. Given a set of data transfer demands between different pairs of nodes, the first problem consists in routing the demands on the ring in either clockwise or counter-clockwise direction so that the maximum data transmitted through any link in either direction is minimized. The second problem, on the other hand, discriminates between the data transmitted in one direction from the other and consists in minimizing the maximum data transmitted in one particular direction through any link. The first problem is referred to as weighted ring

edge-loading problem in the literature, whereas the latter as the weighted ring arc-loading problem. Computational results on the standard benchmark instances show the effectiveness of our approaches on both the problems.

Chapter 8 concludes the thesis by listing the contributions made in addressing the afore-mentioned seven problems. It also provides some guidelines and directions for future research.

Chapter 2

p-median problem

2.1 Introduction

Given an undirected graph $G = (V, E)$ with $|V| = n$, the p-median problem seeks on this graph a set $Z \subseteq V$ of p vertices that minimizes the sum of distances from all the vertices in V to their respective closest vertices in Z. The vertices of the graph and the vertices in Z can be regarded as the demand points and the potential locations of facilities respectively, the objective is to select the locations of p facilities to serve n demand points, so that the sum of the distances of demand points from their nearest facilities is minimized. Throughout this chapter, the vertices in set Z and facility locations are used interchangeably. There are many real world situations which can be modelled using the p-median problem, for example, locating public facilities, industrial plants, and ware-houses. p-median problem can also be represented in a matrix form as follows: Given a matrix D of dimension $n \times n$, select p columns of D in such a way, so that the sum of minimum coefficients in each row within these p columns is as small as possible.

Kariv and Hakimi [75] showed that the p-median problem is $\mathcal{NP}$-hard. Therefore, no polynomial time algorithm exists to solve this problem. The small size instances of p-median problem can be solved within reasonable computational time by exact algorithms but with the increase in problem size, the computation time also increases exponentially. Hence, we need heuristics to solve large instances of the problem. Heuristic algorithms do not guarantee the optimal solution, but a feasible solution can be obtained which is likely to be either optimal or near optimal. Heuristic algorithms work

in acceptable time and memory requirements.

The first heuristic method for the p-median problem was proposed by Kuehn and Hamburger [76]. This method starts with an empty set of facilities and then iteratively adds facilities till p-facilities are added. During each iteration, the location which reduces the objective function value by the maximum amount is chosen for locating a facility. Feldman et al. [77] proposed a stingy heuristic which starts with all n points to be the location of facilities and then iteratively deletes facilities till only p-facilities remain. During each iteration, the facility whose deletion increases the objective function value by the least amount is chosen for deletion. Another early heuristic was proposed by Teitz and Bart [78] which is based on interchanging a facility location with a non-facility location. Since the development of these early heuristics, a number of heuristic and metaheuristic algorithms have been proposed for p-median problem, e.g. fast interchange heuristic [79], Lagrangean relaxation based heuristics [80, 81, 82], simulated annealing [83], tabu search [84, 85], variable neighborhood search [86], genetic algorithms [87, 88, 89, 90], global/regional interchange algorithm [91], LEVEL-2 and LEVEL-3 heuristics [92], gamma heuristic [93], particle swarm optimization [94, 95], hybrid heuristic methods [96], a swap-based local search procedure [97], a parallel genetic algorithm [98], artificial bee colony algorithm [99]. A survey on heuristic and metaheuristic approaches for the p-median problem can be found in [15].

In this chapter, we have proposed an artificial bee colony (ABC) algorithm based approach for the p-median problem. There also exists a previously proposed ABC algorithm for the p-median problem developed by Basti and Sevkli [99]. However, our ABC algorithm is entirely different from the one proposed in [99]. In their approach, Basti and Sevkli [99] used ABC algorithm which works with continuous values and they have to convert continuous values to discrete values in order to get a solution. This has motivated us to develop our algorithm which works with discrete values. In our algorithm, the conversion from continuous values to discrete values is not needed. We have devoted an entire subsection (Section 2.3.7) to highlight the key differences between our approach and approach of [99]. We have compared our ABC approach with some state-of-the-art approaches including ABC approach of [99] on the two sets of standard benchmark instances, viz. OR-Library test instances and Galvao test instances. Computational results show the effectiveness of our approach.

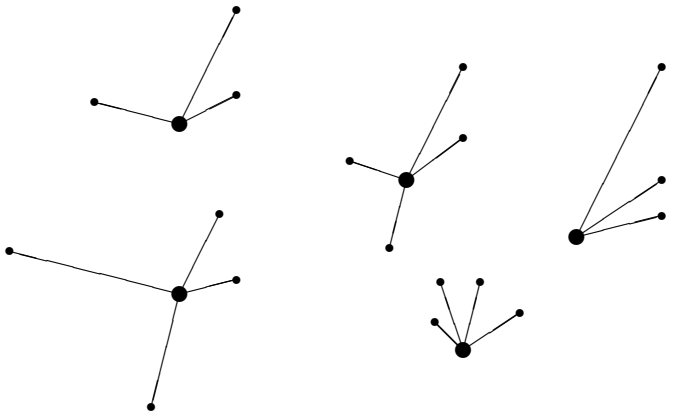


Figure 2.1: A typical solution of the p-median problem with $n = 23$ & $p = 5$

The remaining part of this chapter is organized as follows: Section 2.2 defines the p-median problem formally. Section 2.3 presents our ABC approach to solve the p-median problem. Section 2.4 reports the computational results and compares our approach with other approaches available in the literature. Finally, Section 2.5 outlines some concluding remarks.

2.2 Formal problem definition

The p-median problem can be defined formally as follows. Given an undirected graph $G = (V, E)$ with vertex set $V = \{v_1, v_2, \ldots, v_n\}$, edge set E and the shortest path or distance from vertex v_i to vertex v_j, $d(v_i, v_j)$. Now, the problem is to find a subset $Z \subset V$ containing p vertices of G, in such a way that the sum of distances from all vertices in V to their closest vertices in Z is minimized, i.e., choose $Z = \{z_1, z_2, \ldots, z_p\}$ that minimizes

$$\sum_{i=1}^{n} \min_{j \in \{1,\ldots,p\}} d(v_i, z_j) \tag{2.1}$$

Figure 2.1 illustrates a typical solution of p-median problem with $p = 5$ facilities and $n = 23$ demand points. Each dot (big or small) represents a demand point. Those demand points which are also the location of facilities are represented by big dots. The lines joining the facilities to the demand points provide an idea of distances between demand points and their nearest facility.

2.3 ABC approach for the p-median problem

This section describes our ABC approach for the p-median problem. The salient features of our ABC approach are discussed below in subsequent subsections. The last subsection lists the differences between our ABC approach and ABC approach of Basti and Sevkli [99]. Our proposed ABC approach will be referred to as ABC-D hereafter.

2.3.1 Solution representation and fitness

We have represented a solution directly by the subset of vertices used for locating the facilities. Demand points are always assigned to the nearest facility in the subset for calculating the objective function value (Equation (2.1)). We have also used the objective function as the fitness function, i.e., a lesser value of the fitness function indicates a more fit solution.

2.3.2 Food source selection for onlooker bees

For selecting a food source for an onlooker bee, we have used the binary tournament selection method. In this selection method, two food sources are selected randomly, and their fitness is compared. The better of the two food sources in terms of their fitness is selected with the probability p_{onl}, otherwise the worse of the two food sources is selected (with the probability $1 - p_{onl}$). Algorithm 3 provides the pseudo-code for the binary tournament selection method.

Algorithm 3: Pseudo-code for the binary tournament selection method

```
Select two solutions e_1 and e_2 from the employed bee solutions randomly;
Generate a random number p between 0 and 1;
if p ≤ p_onl then
    return the best solution among e_1 and e_2
else
    return the worst solution among e_1 and e_2
```

2.3.3 Initial solution

The initial solutions for our ABC algorithm are generated in a random manner. One location for a facility is selected at a time randomly from the not yet selected vertices

of V, and this process is repeated until p facilities are located.

2.3.4 Neighboring solution generation

Our neighboring solution generator operator is based on the fact that if a facility is located at a particular demand point in one good solution, then it is highly likely that a facility is located at this demand point in many good solutions. To generate a neighbor solution Z' for solution Z, we select a random solution Z_1 from the population of solutions. Then we copy those locations of facilities which are present in both solutions (common facility locations) Z and Z_1 into Z'. Next, a fraction f of the remaining locations for facilities are added from original solution Z. And the rest of locations for facilities are added from random solution Z_1. The parameter f is determined empirically. While adding locations (from Z and Z_1), we always add that location to solution Z' whose addition yields the smallest objective function value assuming only that many facilities are opened.

If the two solutions Z and Z_1 are identical, i.e., these two solutions contain same locations for all the facilities. In this case copying all the location to Z' will produce another solution identical to Z and Z_1, hence resulting in one more copy of Z (and Z_1). This situation is known as a collision in ABC algorithm jargon [51]. If a collision occurs while generating a neighboring solution for an employed bee, then that solution is discarded, and the concerned employed bee becomes a scout. The scout bee is reinitialized as employed bee by associating it with a new solution generated randomly in a fashion similar to an initial solution. This is done to eliminate one duplicate solution. If a collision occurs while generating a neighboring solution for an onlooker bee then another solution is chosen randomly. This process of selecting a random solution is repeated until a solution different from original solution is found for an onlooker bee. Instead of tackling the collision like employed bee phase, the reason for repeatedly searching for a solution different from original solution for an onlooker lies in the fact that it is of no use to generate a solution randomly for an onlooker bee as such a solution can seldom survive. An onlooker bee solution can survive only when it is better than the original solution with which the onlooker in consideration is associated and solutions of all other onlooker bees which are also associated with the same original solution. A randomly generated solution can seldom be better than all these solutions.

2.3.5 Other features

An employed bee abandons a solution when there is no improvement in the quality of this solution over a specified number of iterations say *limit* and becomes a scout. For ABC algorithm, *limit* is an important control parameter and should be appropriately set as it is responsible for maintaining the delicate balance between exploration and exploitation. A lesser value of limit favors exploration over exploitation. On the other hand, a higher value of limit gives preference to exploitation undermining exploration. An employed bee can also become a scout, as mentioned in Section 2.3.4 through a collision.

A newly designated scout bee is immediately associated with a newly generated food source and it becomes employed again. This food source is generated randomly in the same manner as an initial solution. In our algorithm, there is no explicit upper and lower limits on the number of scouts in a single iteration. The number of scouts in a particular iteration depends on the above mentioned two conditions.

2.3.6 Local search

This local search is applied after the execution of ABC algorithm on the best solution obtained through ABC algorithm only. In this local search, each vertex z in a solution Z is tried to be exchanged one-by-one with a vertex not in Z so that the value of the objective function is reduced by the largest amount. This process is repeated till the objective function can not be improved further.

Algorithm 4 provides the pseudo-code of our ABC approach where *generate_neighbor*(Z) is a function that take as input a solution Z and returns a solution in the neighborhood of Z (first paragraph in Section 2.3.4) and *local_search*(Z) is a function that take a solution Z as input and return a solution obtained after applying local search on it (Section 2.3.6). *binary_tournament*$(e_1, e_2, \ldots, e_{n_e})$ is another function that selects a solution among employed bee solutions $e_1, e_2, \ldots, e_{n_e}$ using binary tournament selection method (Section 2.3.2) and returns the index of the solution selected.

2.3.7 Key points of difference with a related work

Basti and Sevkli [99] have proposed another ABC algorithm for the p-median problem. This subsection compares our approach with them. The key differences are summarized

Algorithm 4: Pseudo-code of our ABC algorithm

```
Randomly generate n_e employed bee solutions e_1, e_2 ..., e_{n_e};
best_sol := best solutions among e_1, e_2, ..., e_{n_e};
while termination condition is not satisfied do
    for i := 1 to n_e do
        e' := generate_neighbor(e_i);
        if e' is better than e_i then
            e_i := e'
        if e' is better than best_sol then
            best_sol := e';
    for i := 1 to n_o do
        k_i := binary_tournament(e_1, e_2, ..., e_{n_e});
        onl_i := generate_neighbor(e_{k_i});
        if onl_i is better than best_sol then
            best_sol := onl_i;
        if onl_i is better than e_{k_i} then
            e_{k_i} := onl_i;
    for i := 1 to n_e do
        if e_i has not improved over last limit iterations then
            replace e_i with a random solution;
local_search(best_sol);
return best;
```

below:

- Basti and Sevkli [99] have used a real vector of length n to encode a solution in their ABC algorithm. To decode a solution from this real vector, indices corresponding to smallest p values in this vector are found, and demand points corresponding to these indices are assumed to be the location of facilities. On the other hand, in our ABC algorithm, we have represented a solution directly by the subset of vertices used for locating the facilities, hence the length of a solution is equal to p ($p < n$). The encoding scheme of [99] suffers from the problem of redundancy, i.e., the same solution can be encoded in many different ways. In

fact, in the encoding scheme of [99], each solution can be represented in infinitely many ways. As ABC algorithm works in the space of encoded solutions, in the presence of redundancy, it has to search a larger space which can severely impair its performance. On the other hand, the encoding scheme used by us does not suffer from the problem of redundancy as each solution is represented uniquely. The size of search space in [99] is infinite, whereas in our case it is $\binom{n}{p}$. Besides, the length of an encoded solution has an adverse impact on the efficiency of several operators associated with ABC algorithm. As $p < n$, so on this aspect also our encoding is better.

- Real vector encoding scheme of [99] incurs a decoding overhead to get the actual solution from its encoded version. In our case, no decoding overhead is incurred as each solution is represented directly by the subset of vertices used for locating the facilities.

- As Basti and Sevkli [99] used a real vector to encode a solution, they followed the original neighboring solution generation method proposed by Karaboga [46], i.e., a neighboring solution is generated by changing the value of one randomly chosen parameter of the original solution. On the other hand, our neighboring solution generation method is based on the assumption that if a vertex is present in one good solution then there are chances that the same vertex may appear in many good solutions. Hence, we have given maximum attention to those vertices which are common in original solution and randomly selected solution, followed by those vertices which are in one of these solutions.

- Basti and Sevkli used roulette wheel selection method for selecting a solution for an onlooker bee. We have used binary tournament selection method for selecting a solution for an onlooker bee. It is an established fact that binary tournament selection method performs better than roulette wheel selection method and at the same time its computationally less expensive. In fact, it has roughly the same performance as rank selection method [60].

- In their work, a greedy local search algorithm is applied on every solution generated by the ABC algorithm. While applying this local search on a solution,

facility locations are considered one-by-one. The facility location in consideration is tried for replacement with all other non-facility locations. The location that yields the least objective function value is retained and then the next facility location is considered. Instead of applying the local search on every solution generated through ABC algorithm, in our work, the local search is applied only on the best solution obtained after the execution of ABC algorithm. Our local search is similar to the local search of Basti and Sevkli except for the fact that we keep applying this local search repeatedly as long as there is an improvement in solution quality, i.e., our local search stops when a complete pass through the existing facility locations fails to improve the solution quality.

Hereafter, the ABC approach of Basti and Sevkli [99] will be referred to as ABC-BS.

2.4 Computational results

Our ABC-D approach has been implemented in C and executed on a Linux based Intel Core i5 2400 system with 4 GB RAM running at 3.10 GHz. In all our computational experiments, the number of employed bees (n_e) is taken to be 50, the number of onlooker bees (n_o) is taken to be 100, p_{onl} is set to 0.75, $limit$ is set to 50. Our ABC-D approach terminates after 100 iterations. All these parameter values were chosen empirically after a large number of trials.

We have compared the performance of ABC-D with the following approaches – particle swarm optimization (PSO) [94], novel discrete particle swarm optimization (NDPSO) [95], tabu search (TS) [85], simulated annealing (SA) [83], genetic algorithm (GA) [90], artificial bee colony algorithm (ABC-BS) [99]. PSO, NDPSO, TS, SA, and GA approaches were also considered in [99] for comparing the performance of ABC-BS. Like [99], we have executed our approach 30 independent times on each test instance and used the same two performance measures, viz. the relative percentage of error (RPE) and fMedian RPE to asses the performance of an approach over a test instance. These two measures are defined as below:

$$\text{RPE} = \frac{f_{min} - f_{opt/best}}{|f_{opt/best}|} \times 100$$

$$\text{fMedian RPE} = \frac{f_{median} - f_{opt/best}}{|f_{opt/best}|} \times 100$$

where f_{min} & f_{median} are the best and median results obtained over 30 runs on a test instance under investigation and $f_{opt/best}$ is the optimal or best known solution value on the same test instance.

Same two sets of test instances as used in [99] are used here to evaluate the performances of different approaches. The first set of test instances is from OR-Library [1], where these instances are listed as instances of uncapacitated p-median problem. In this set, there are 40 problem instances varying in size and difficulty. The optimum solutions for all the instances in this set are also known. These instances are the most widely used test instances for p-median problem. The second set of test instances on which we conducted our experiments is known as Galvao test instances. This problem set was introduced by Galvao and ReVelle [100], and it consists of 16 problem instances. The Galvao test instances are relatively smaller in size than the test instances of OR-Library.

Please note that the results of PSO, NDPSO, TS, SA, GA, and ABC-BS on the aforementioned two sets of test instances are taken from [99] where fMedian RPE is reported for PSO and ABC-BS only. Further, on Galvao test instances only GA, PSO, and ABC-BS were compared in terms of RPE, whereas for OR-Library instances all the approaches are compared in terms of RPE. Accordingly, we have reported the results of these approaches.

2.4.1 Performance comparison on OR-Library test instances

Table 2.1 presents the results of various approaches on 40 OR-Library instances. This table reports for each instance, the RPE values for ABC-D, PSO, NDPSO, TS, SA, GA & ABC-BS and the fMedian RPE values for ABC-D, PSO & ABC-BS. It also reports the number of instances optimally solved out of 40 by each approach and the average RPE & average fMedian RPE values achieved over 40 instances for various approaches. The cases where ABC-D obtained better value than all the other methods are reported in bold font. These results clearly show the superiority of our approach over other approaches. ABC-D is able to find the optimal solution on 31 instances out of 40, whereas ABC-BS finds 30 optimal solutions values. All other approaches solve the lesser number of instances optimally. Overall, the average RPE and average

[1] `http://www.brunel.ac.uk/~mastjjb/jeb/info.html`

fMedian RPE values are least among all approaches. This indicates that when ABC-D is not able to solve an instance optimally, solutions returned by it are quite close to optimal solution values. The reported results show that our proposed approach obtained optimal values for instances with small p value. And small deviations from the optimal values are obtained for instances with greater p value. Except for small values of n, this happens with every other value of n. However, in these cases also, we reach closer to the optimal values when compared to most other methods. In fact, every approach considered here find instances with large p values difficult. This difficulty can be explained theoretically also on the basis of search space size. Actually, there are *nchoosep* solutions and for the fixed value of n, the number of solutions increase with increase in p till $p = \lfloor \frac{n}{2} \rfloor$. All the values of p in the instances considered here is less than $\lfloor \frac{n}{2} \rfloor$, and hence, the search space size increases with increase in p for fixed n. As the approaches have to search a larger search space, they find these instances relatively difficult.

2.4.2 Performance comparison on Galvao test instances

Table 2.2 presents the results of various approaches on 16 Galvao test instances. The format of this table is similar to Table 2.1 except for the fact that the RPE values are reported for ABC-D, PSO, GA & ABC-BS and the fMedian RPE values are reported for ABC-D, PSO & ABC-BS. On these instances, our results are much better than other approaches. Our approach reached the optimum solution value for 13 out of 16 instances, whereas GA, PSO, and ABC-BS respectively found optimal solutions for 9, 4, and 5 problems out of 16. Our RPE and fMedian RPE values are also much less in comparison to other approaches on most of the instances.

2.4.3 Comparison of execution times taken by various approaches

In this section, we have reported the execution times of our approach on both the data sets and compared them with those approaches for which execution times were reported in their respective papers. Only for NDPSO [95], SA [83], GA [90], execution times were reported. Among these three methods, first two methods were tested on OR-Library instances only, whereas GA was executed on both the data sets. Hence, for Galvao test instances, only the execution times of GA is available.

Table 2.3 and Table 2.4 report the execution times of various approaches for OR-Library and Galvao test instances respectively. The data for NDPSO, SA, and GA is taken from their respective papers. The ABC-BS, NDPSO, SA, and GA were executed 30, 10, 100 and 10 independent times respectively. The times reported in these tables for various methods are average execution times over their respective runs. For SA, [83] reports the time per cycle and we have multiplied it with the total number of cycles to get the execution time for each instance. NDPSO was executed on a 2.6GHz PC with 512MB memory, SA was executed on an RISC/6000 workstation and GA was executed on a 733 MHz Pentium 3 system with 128 MB RAM. As NDPSO, SA, and GA approaches were executed on systems which are different from the system used to execute ABC-D, hence execution times can not be compared precisely, and only a rough comparison can be possible [101]. In Table 2.3, columns 1,2 and 3 represent instance number, the number of nodes and the number of centers and the remaining four columns report the execution times of NDPSO [95], SA [83], GA [90], and ABC-D.

From this table, it is clear that our ABC-D approach is faster than NDPSO even after compensating for differences in processing speeds on most of the instances. On the other hand, GA is faster than our approach on most of the instances. However, not much can be said about the relative performance of SA, and ABC-D with certainty. SA seems to be faster on some instances, whereas ABC-D on some others. In Table 2.4 also, columns 1, 2 and 3 represent instance number, the number of nodes and number of centers respectively, and the remaining two columns report the execution times of GA [90] and ABC-D. From this table, it is clear that GA is faster than ABC-D on Galvao test instances. Execution times were not reported in [99], and hence, we can not compare the execution times of ABC-D with those of ABC-BS. However, considering the fact that ABC-BS applies the local search on each solution generated and uses a real vector encoding scheme which incurs a decoding overhead, it can not be faster than ABC-D.

2.4.4 Impact of parameter settings, and convergence behavior

In order to study the impact of parameter settings on solution quality, we have taken 4 different test instances, viz. Galvao8, pmed20, pmed25, and pmed5. The value of each parameter is varied one by one while keeping the values of all other parameters fixed to their respective values mentioned at the beginning of this section. The results are

Table 2.1: Performance on OR-library test instances Results

				RPE							fMedian RPE		
Problem	n	p	optimal	PSO	NDPSO	SA	GA	TS	ABC-BS	ABC-D	PSO	ABC-BS	ABC-D
pmed1	100	5	5819	0	0	0	0	0	0	0	0	0	0
pmed2	100	10	4093	0	0.15	0	0	0	0	0	0.29	0	0.08
pmed3	100	10	4250	0	0	0	0	0	0	0	0	0	0
pmed4	100	20	3034	0	0	0	0	0	0	0	0.73	0	0.06
pmed5	100	33	1355	0	0.11	0	0	0	0	0	0.15	0	0.02
pmed6	200	5	7824	0	0	0	0	0	0	0	0	0	0
pmed7	200	10	5631	0	0	0	0	0	0	0	0	0	0
pmed8	200	20	4445	0	0	0	0	0.2	0	0	0.2	0	0.05
pmed9	200	40	2734	0	0.24	0	0	0.47	0	0	0.11	0.15	0.15
pmed10	200	67	1255	0	0.12	0	0.08	0.08	0	0	0.88	0.72	**0.1**
pmed11	300	5	7696	0	0	0	0	0	0	0	0	0	0
pmed12	300	10	6634	0	0	0	0	0	0	0	0	0	0
pmed13	300	30	4374	0	0	0	0	0	0	0	0.16	0	0
pmed14	300	60	2968	0.03	0.17	0.47	0	0	0.11	0	0.13	0.27	0.15
pmed15	300	100	1729	0.23	0.27	0.35	0.23	0.63	0.16	**0.06**	0.64	0.46	**0.19**
pmed16	400	5	8162	0	0	0	0	0	0	0	0	0	0
pmed17	400	10	6999	0	0.02	0	0	0	0	0	0	0.06	0.02
pmed18	400	40	4809	0.04	0.18	0.27	0	0.13	0	0.04	0.46	0.12	**0.08**
pmed19	400	80	2845	0.18	0.66	0.77	0.04	0	0.04	0.07	0.56	0.11	0.19
pmed20	400	133	1789	0	0.92	0.11	0.17	0.28	0	0	0.45	0.11	0.33
pmed21	500	5	9138	0	0	0	0	0	0	0	0	0	0
pmed22	500	10	8579	0	0	0	0	0	0	0	0	0	0
pmed23	500	50	4619	0	0.68	0	0	0	0	0	0	0	0.08

pmed24	500	100	2961	0	0.97	0.14	0.03	0.17	0	0	0.37	0.03	0.21
pmed25	500	167	1828	0.49	0.95	0.16	0.22	0.33	0.26	0.16	0.98	1.2	**0.45**
pmed26	600	5	9917	0	0	0	0	0	0	0	0	0	0
pmed27	600	10	8307	0	0.01	0	0	0.04	0	0	0.04	0	0.01
pmed28	600	60	4498	0.09	0.92	0.09	0.02	0.11	0.02	**0**	0.11	0.02	0.09
pmed29	600	120	3033	0.1	0.96	0.2	0.07	0.3	0.17	**0.03**	0.36	0.3	**0.15**
pmed30	600	200	1989	0.45	0.93	0.15	0.4	0.4	0.33	0.4	1.31	0.8	**0.62**
pmed31	700	5	10086	0	0	0	0	0	0	0	0	0	0
pmed32	700	10	9297	0	0	0	0	0	0	0	0.04	0	0
pmed33	700	70	4700	0.04	0.94	0.04	0	0.02	0.02	0	0.26	0.45	**0.12**
pmed34	700	140	3013	0.23	0.97	0.4	0.07	0.13	0.03	0.03	0.46	0.23	0.26
pmed35	800	5	10400	0	0	0	0	0	0	0	0	0	0
pmed36	800	10	9934	0	0.12	0	0	0	0	0	0	0.17	0.04
pmed37	800	80	5057	0.08	0.94	0.1	0.02	0.12	0	0.02	0.18	0.12	**0.08**
pmed38	900	5	11060	0	0	0	0	0	0	0	0	0	0
pmed39	900	10	9423	0	0	0	0	0	0	0	0	0	0
pmed40	900	90	5128	0.04	0.97	0.16	0.1	0.14	0.1	0.06	0.18	0.27	0.22
Average				0.05	0.305	0.085	0.036	0.089	0.031	**0.022**	0.226	0.14	**0.094**
Number of problems solved optimally				28	18	26	28	24	30	**31**			

Table 2.2: Performance comparison on Galvao test instances

				RPE				fMedian RPE		
Instance	n	p	Optimal	GA	PSO	ABC-BS	ABC-D	PSO	ABC-BS	ABC-D
Galvao1	100	5	5703	0	0	0	0	0	0	0
Galvao2	100	10	4426	0.32	0.32	0.27	**0**	3.03	2.49	**0.01**
Galvao3	100	15	3893	0	0	0	0	2.62	0.28	**0**
Galvao4	100	20	3565	0	0.17	0.21	0	0.2	0.28	**0.01**
Galvao5	100	25	3291	0.03	0.03	0.03	**0**	0.79	0.4	**0.03**
Galvao6	100	30	3032	0	0.07	0.09	0	0.33	0.43	**0.05**
Galvao7	100	35	2784	0	0.18	0.19	0	0.18	0.22	**0**
Galvao8	100	40	2542	0	0.2	0	0	0.24	0	0.01
Galvao9	150	5	10839	0	0	0	0	0.02	0.02	**0**
Galvao10	150	15	7390	0	0.41	0	0	1.54	0.14	**0.09**
Galvao11	150	20	6454	0.12	0	0.21	0	2.62	1.18	**0.28**
Galvao12	150	25	5875	0	0.51	0.23	0	0.51	1.89	**0.32**
Galvao13	150	35	5192	0.04	0.13	0.16	0.04	0.81	0.6	**0.11**
Galvao14	150	45	4636	0.11	0.15	0.19	**0.02**	0.56	0.47	**0.1**
Galvao15	150	50	4374	0.14	0.18	0.07	**0**	0.37	0.21	**0.06**
Galvao16	150	60	3873	0.08	0.21	0.08	**0.05**	0.28	0.1	**0.06**
Average				0.053	0.16	0.108	**0.007**	0.881	0.544	**0.071**
Number of problems solved optimally				9	4	5	**13**			

Table 2.3: Execution times of various approaches on OR-library test instances

			Time(in seconds)			
Instance	n	p	NDPSO	SA	GA	ABC-D
1	100	5	0.56	5.00	1.00	0.92
2	100	10	21.60	9.00	1.00	2.26
3	100	10	2.02	9.00	2.00	2.20
4	100	20	11.84	15.00	2.00	3.82
5	100	33	33.79	21.00	3.00	5.72
6	200	5	1.37	20.00	4.00	2.28
7	200	10	18.72	36.00	5.00	4.81
8	200	20	21.06	66.00	7.00	9.59
9	200	40	119.44	114.50	12.00	18.39
10	200	67	105.80	158.00	20.00	27.74
11	300	5	2.04	20.60	17.00	3.30
12	300	10	5.13	39.40	12.00	7.99
13	300	30	42.93	106.40	21.00	30.77
14	300	60	144.05	186.40	44.00	63.18
15	300	100	150.24	256.00	63.00	99.48
16	400	5	6.36	107.20	23.00	4.95
17	400	10	57.98	217.20	24.00	10.79
18	400	40	150.48	500.40	56.00	62.68
19	400	80	158.84	868.00	133.00	137.80
20	400	133	293.66	598.40	163.00	222.87
21	500	5	4.24	57.20	38.00	6.01
22	500	10	33.00	56.00	45.00	14.83
23	500	50	155.87	241.00	159.00	124.46
24	500	100	394.75	441.00	211.00	272.11
25	500	167	727.81	585.00	316.00	448.05
26	600	5	20.79	44.00	68.00	8.25
27	600	10	74.44	80.00	78.00	19.28
28	600	60	260.62	434.00	245.00	210.91
29	600	120	708.85	780.00	437.00	451.63
30	600	200	1972.30	1070.00	790.00	734.07
31	700	5	53.87	63.00	145.00	9.21
32	700	10	77.96	123.00	132.00	22.78
33	700	70	451.27	736.00	454.00	299.81
34	700	140	1303.58	1322.00	652.00	701.22
35	800	5	25.38	82.00	156.00	10.67
36	800	10	113.79	158.00	185.00	26.21
37	800	80	948.08	1108.00	759.00	484.96
38	900	5	29.19	107.00	288.00	12.47
39	900	10	111.10	207.00	265.00	30.11
40	900	90	1393.01	1569.00	1322 00	635.47

Table 2.4: Execution times of various approaches on Galvao test instances

			Time(in seconds)	
Instance	n	p	GA	ABC-D
Galvao1	100	5	1.00	1.04
Galvao2	100	10	1.00	2.13
Galvao3	100	15	2.00	2.94
Galvao4	100	20	2.00	3.80
Galvao5	100	25	2.00	4.59
Galvao6	100	30	2.00	5.65
Galvao7	100	35	3.00	6.16
Galvao8	100	40	3.00	6.13
Galvao9	150	5	2.00	1.57
Galvao10	150	15	5.00	5.51
Galvao11	150	20	6.00	7.22
Galvao12	150	25	6.00	8.37
Galvao13	150	35	9.00	10.87
Galvao14	150	45	9.00	13.27
Galvao15	150	50	9.00	14.09
Galvao16	150	60	12.00	14.94

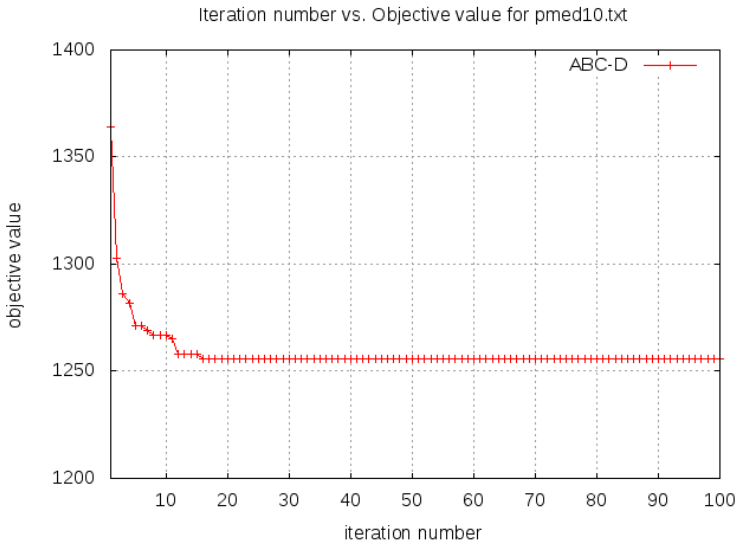


Figure 2.2: Convergence behavior of ABC-D on pmed10 test instance

Table 2.5: Influence of parameter settings on solution quality

Parameter	Value	Galvao8		pmed5		pmed17		pmed32	
		BestF	AvgF	BestF	AvgF	BestF	AvgF	BestF	AvgF
iterations	50	2542.00	2542.33	1355.00	1355.50	6999.00	7004.03	9297.00	9297.80
	100	**2542.00**	**2542.13**	**1355.00**	**1355.20**	**6999.00**	**6999.13**	**9297.00**	**9297.00**
	150	2542.00	2542.07	1355.00	1355.10	6999.00	6999.13	9297.00	9297.13
	200	2542.00	2542.07	1355.00	1355.20	6999.00	6999.13	9297.00	9297.00
limit	25	2542.00	2542.10	1355.00	1355.20	6999.00	7000.73	9297.00	9297.13
	50	**2542.00**	**2542.13**	**1355.00**	**1355.20**	**6999.00**	**6999.13**	**9297.00**	**9297.00**
	75	2542.00	2542.17	1355.00	1355.00	6999.00	6999.93	9297.00	9297.27
	100	2542.00	2542.17	1355.00	1355.20	6999.00	7000.07	9297.00	9297.53
n_e	25	2542.00	2542.30	1355.00	1355.30	6999.00	7000.83	9297.00	9297.80
	50	**2542.00**	**2542.13**	**1355.00**	**1355.20**	**6999.00**	**6999.13**	**9297.00**	**9297.00**
	75	2542.00	2542.13	1355.00	1355.00	6999.00	6999.93	9297.00	9297.27
	100	2542.00	2542.00	1355.00	1355.00	6999.00	6999.53	9297.00	9297.00
n_o	50	2542.00	2542.37	1355.00	1355.20	6999.00	7000.73	9297.00	9297.40
	75	2542.00	2542.23	1355.00	1355.20	6999.00	7000.53	9297.00	9297.13
	100	**2542.00**	**2542.13**	**1355.00**	**1355.20**	**6999.00**	**6999.13**	**9297.00**	**9297.00**
	125	2542.00	2542.10	1355.00	1355.50	6999.00	7000.07	9297.00	9297.40
	150	2542.00	2542.10	1355.00	1355.20	6999.00	6999.93	9297.00	9297.00
p_{onl}	0.65	2542.00	2542.13	1355.00	1355.30	6999.00	7000.73	9297.00	9297.13
	0.7	2542.00	2542.10	1355.00	1355 .00	6999.00	6999.93	9297.00	9297.00
	0.75	**2542.00**	**2542.13**	**1355.00**	**1355.20**	**6999.00**	**6999.13**	**9297.00**	**9297.00**
	0.8	2542.00	2542.17	1355.00	1355.40	6999.00	7000.97	9297.00	9297.53
	0.85	2542.00	2542.33	1355.00	1355.30	6999.00	7000.97	9297.00	9297.53

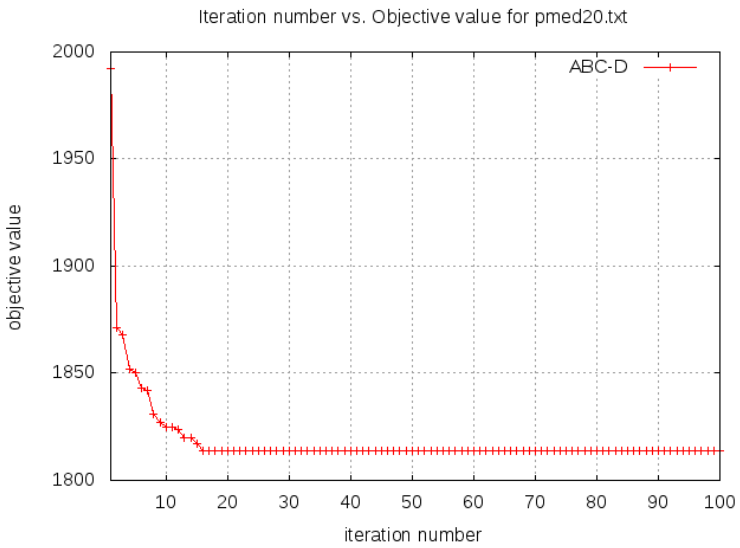


Figure 2.3: Convergence behavior of ABC-D for pmed20 test instance

reported in Table 2.5. Values in bold in this table indicate the results with original parameter values which are used in ABC-D for all the experiments. From this table, it can be clearly seen that the values chosen by us provide either the optimal results or results which are very near to optimal results. In those cases where we have not got the best results with chosen parameter values, the parameter values chosen have provided best results on some other instances not included in this table.

To show the convergence behavior of ABC-D, we have chosen 2 test instances of different sizes, viz. pmed10, and pmed20. The convergence behavior of ABC-D for pmed10, and pmed20 is shown in Figure 2.2 and Figure 2.3 respectively. These figures clearly demonstrate that ABC-D converges rapidly to best solution values.

2.5 Conclusions

In this chapter, an artificial bee colony algorithm based approach is proposed for solving the p-median problem. We have compared the performance of our approach against some state-of-the-art approaches available in the literature on two widely used p-median benchmark sets, viz. OR-Library test instances and Galvao test instances. Computational results show the superiority of our approach in comparison to other approaches. The performance of our approach depends on the number of facilities p to be opened.

Our approach reached the optimum solution values for the cases where the p value is small. However, we obtained solution values slightly worse than optimal for those instances where the p value is large. A theoretical justification is provided for this observation.

Chapter 3

p-median problem with positive/negative weights

3.1 Introduction

This chapter is concerned with an extension of the p-median problem considered in the previous chapter, where each vertex of the graph is associated with a weight. This weight can be either positive or negative. A location problem with positive and negative weights on the vertices is useful in applications, where some facilities are non-attractive to some clients (facilities are obnoxious). Many obnoxious location problems are discussed and classified in [38]. Interested readers can find the survey on obnoxious location problems in [102] and [103].

Burkard [104] proposed the first location model with positive and negative weights and also proved that the 1-median problem on a cactus with positive and negative vertex weights can be solved in linear time. Burkard et al. [105] observed that there exist, two different models when p-median problem with positive/negative weights in graphs is considered. These two models are described here by following the same notational conventions as used in the previous chapter. In the first model, referred to as P1, the objective is to minimize the sum of the minimum weighted distances, i.e.,

$$\sum_{i=1}^{n} \min_{j \in \{1,\ldots,p\}} (w_i d(v_i, z_j)) \tag{3.1}$$

In the second model, referred to as P2, the objective is to minimize the sum of weighted minimum distances, i.e.,

$$\sum_{i=1}^{n} w_i \min_{j \in \{1,\ldots,p\}} d(v_i, z_j) \tag{3.2}$$

Both the models are equivalent when the vertices are associated with positive weights only.

For model P1, Burkard et al. [105] developed an $\mathcal{O}(n^2)$ algorithm for the 2-median problem on a tree. They also developed an $\mathcal{O}(n \log n)$ algorithm for stars and an $\mathcal{O}(n)$ algorithm for paths for first model (P1). They presented an $\mathcal{O}(n^3)$ algorithm for the 2-median problem on a tree for the second model (P2) and showed that the complexity can be reduced to $\mathcal{O}(n^2)$ if the medians are restricted to vertices. Burkard and Fathali [106] presented an algorithm for 3-median problem on a tree for second model (P2). There exist some heuristic methods also to solve the positive/negative weighted p-median problem on graphs under both the models. Fathali and Kakhki[107] developed a modified variable neighborhood search (MVNS). Fathali et al. [108] presented an ant colony optimization algorithm (ACO). A genetic algorithm (GA) for the positive/negative weighted p-median problem is proposed by Fathali [109]. ACO and GA are the two best approaches known so far for solving the positive/negative weighted p-median problem.

In this chapter, we have extended the ABC based approach developed in the previous chapter for the p-median problem to the positive/negative weighted p-median problem. We have compared our ABC approach with ACO [108] and GA [109] on the standard benchmark instances for the problem. Computational results show the effectiveness of our approach.

The remainder of this chapter is organized as follows: Section 3.2 describes our approach based on ABC algorithm to solve the positive/negative weighted p-median problem. Section 3.3 presents the computational results and their analysis. Finally, Section 3.4 provides some concluding remarks.

3.2 ABC approach for the p-median problem with positive/negative weights

This section presents salient features of our ABC approach.

3.2.1 Solution representation, fitness, and initial solution

We have used the same solution representation scheme as used in Chapter 2, i.e., a solution is represented directly by the subset of p vertices where facilities are located. Here also, the objective function is used as the fitness function. So for model P1, fitness is determined using Equation (3.1), whereas, for model P2, fitness is determined using Equation (3.2). Please note the less value of the fitness function means a more fit solution. The two models will differ in the assignment of demand points to the facilities. In model P1, vertices with positive weights are assigned to the nearest facility and vertices with negative weights are assigned to the farthest facility. On the other hand, in model P2, vertices with both positive and negative weights are assigned to the closest facility.

Each initial solution is generated in the same manner as in the approach described in the previous chapter.

3.2.2 Probability of selecting a food source

Like the previous chapter, we have used the binary tournament selection method for selecting a food source for an onlooker where the candidate with better fitness is selected with probability p_{onl}.

3.2.3 Neighboring solution generation

A neighboring solution for our ABC algorithm is generated in the same manner as described in Section 2.3.4 of Chapter 2. However, to improve the quality of the neighboring solution generated, we have used a 1-interchange heuristic at the end of neighboring solution generation process. In 1-interchange heuristic method, we replace one vertex in the solution Z' by a vertex which is not present in it and which results in the maximum reduction in objective function value. We randomly select one vertex in Z' and find a vertex in $\{V - Z'\}$, which results in maximum reduction in objective function value. This method is computationally expensive, because of a large number of fitness calculations performed each time it is applied. However, it aids more often than not in improving the quality of a solution. To balance the computational cost and degree of improvement, we have applied 1-interchange heuristic K times on every solution, where K is a parameter to be determined empirically.

3.2.4 Other features

These features are same as described in Section 2.3.5 of Chapter 2.

3.2.5 Local search

We have used the same local search method as described in Section 2.3.6 of Chapter 2. But, instead of applying this local search only on the best solution found by ABC algorithm, we have applied it on the L best solutions found by ABC algorithm where L is a parameter to be determined empirically.

Algorithm 5 provides the pseudo-code of our hybrid ABC approach where generate_neighbor(Z), 1-interchange(Z) and local_search(Z) are three functions that take as input a solution Z and return respectively a solution in the neighborhood of Z (first paragraph in Section 3.2.3), a solution obtained after applying 1-interchange heuristic on Z (Section 3.2.3), a solution obtained after applying local search on solution Z (Section 3.2.5). binary_tournament_seletion($e_1, e_2, \ldots, e_{n_e}$) is another function that selects a solution among employed bee solutions $e_1, e_2, \ldots, e_{n_e}$ using binary tournament selection method (Section 3.2.2) and returns the index of the solution selected.

3.3 Computational results

Our hybrid ABC approach has been implemented in C and executed on a Linux based Intel Core i5 2400 system with 4 GB RAM running at 3.10 GHz. In all our computational experiments, the number of employed bees (n_e) is taken to be 25, the number of onlooker bees (n_o) is taken to be 50, p_{onl} is set to 0.85, *limit* is set to 50, F_r is set to $\frac{2}{3}$, K is set to 2 and L is set to 5. Our hybrid ABC approach terminates after 100 iterations. All these parameter values were chosen empirically after a large number of trials.

We have compared our hybrid ABC approach with the two best approaches, viz. GA [109] and ACO [108] on both the models. For this comparison, we have used the same 40 test instances as used in [109] and [108]. These instances are slightly modified version of the OR-Library instances used in the previous chapter. A slight modification is done in these instances to accommodate negative weights. Vertices weights in these instances are restricted to ± 1 only. Further, the negative weight of -1 is assigned to only the first 2 or first 5 or first 10 vertices and to all odd numbered vertices. The case

Algorithm 5: Pseudo-Code of our hybrid ABC algorithm

```
Randomly generate n_e employed bee solutions e_1, e_2, ..., e_{n_e};
for i := 1 to L do
    best_sol_i := i^th best solutions among e_1, e_2, ..., e_{n_e};
while termination condition is not satisfied do
    for i := 1 to n_e do
        e' := generate_neighbor(e_i);
        for j = 1 to K do
            e' := 1-interchange(e')
        if e' is better than e_i then
            e_i := e';
        for j = 1 to L do
            if e' is better than best_sol_j then
                best_sol_j := e';
                break;
    for i := 1 to n_o do
        k_i := binary_tournament(e_1, e_2, ..., e_{n_e});
        onl_i := generate_neighbor(e_{k_i});
        for j = 1 to K do
            onl_i := 1-interchange(onl_i)
        for j := 1 to L do
            if onl_i is better than best_sol_j then
                best_sol_j := onl_i;
                break;
    for i = 1 to n_o do
        if onl_i is better than e_{k_i} then
            e_{k_i}:=onl_i
    for i = 1 to n_e do
        if e_i has not improved over last limit iterations then
            replace e_i with a random solution;
for i := 1 to L do
    best_sol_i := local_search(best_sol_i);
best := best solution among best_sol_1, best_sol_2, ..., best_sol_L;
return best;
```

Table 3.1: The average total CPU times (in seconds) for problems P1 and P2

			P1				P2			
SNo	n	p	GA	ABC	ACO	ABC	GA	ABC	ACO	ABC
1	100	5	1.000	0.896	1.280	0.892	1.000	0.831	1.000	0.827
2	100	10	1.550	0.871	1.520	0.872	1.500	0.807	1.400	0.809
3	100	10	1.333	0.873	1.400	0.873	1.500	0.817	1.200	0.815
4	100	20	1.777	0.847	1.160	0.859	1.777	0.821	0.880	0.818
5	100	33	1.777	0.894	0.960	0.910	1.555	0.858	0.640	0.856
6	200	5	2.666	3.697	3.000	3.745	2.444	3.736	2.600	3.734
7	200	10	3.000	3.180	3.200	3.233	2.888	3.363	2.600	3.365
8	200	20	4.777	3.194	2.840	3.243	4.111	3.278	2.640	3.279
9	200	40	7.222	3.377	2.760	3.444	5.777	3.455	2.440	3.455
10	200	67	8.222	3.792	2.360	3.703	6.777	3.742	1.720	3.859
11	300	5	9.777	8.085	10.480	7.975	7.777	8.124	10.192	8.064
12	300	10	11.888	7.804	10.640	7.782	11.666	7.876	9.238	7.802
13	300	30	21.777	7.809	10.730	7.827	18.444	7.605	11.572	7.535
14	300	60	38.625	8.979	11.530	8.845	36.125	9.070	10.411	8.503
15	300	100	35.777	9.537	11.819	9.607	32.444	9.069	9.525	9.027
16	400	5	25.333	15.307	27.787	15.421	24.777	15.447	25.238	15.330
17	400	10	41.500	13.844	26.512	13.823	37.000	13.857	21.400	13.701
18	400	40	63.888	14.278	26.231	14.153	58.111	13.548	20.000	13.295
19	400	80	105.777	17.209	26.852	16.649	86.333	15.789	28.079	15.979
20	400	133	151.555	20.514	32.183	19.595	117.888	18.000	25.911	18.048
21	500	5	49.777	25.129	46.960	25.157	43.375	23.656	35.400	23.577
22	500	10	80.777	22.983	48.957	22.827	75.666	22.769	45.755	22.777
23	500	50	143.333	24.257	50.299	23.981	127.111	22.172	39.292	22.039
24	500	100	242.000	28.925	53.628	29.082	218.222	25.758	43.935	25.735
25	500	167	363.222	35.381	63.856	35.769	279.444	30.781	60.158	31.098
26	600	5	102.000	37.861	74.440	38.055	102.222	39.522	69.460	39.227
27	600	10	165.666	34.483	72.760	34.330	141.666	36.340	61.400	36.351
28	600	60	336.777	37.419	87.052	36.618	284.888	37.198	68.514	36.889
29	600	120	540.444	49.204	100.343	48.915	531.888	47.151	77.294	47.486
30	600	200	817.111	55.105	112.216	56.410	734.000	57.405	83.340	57.224
31	700	5	162.777	55.871	116.082	55.856	141.429	69.401	87.480	68.134
32	700	10	246.333	51.196	128.334	51.373	201.111	63.564	92.130	63.109
33	700	70	607.222	59.753	152.155	57.710	558.555	70.251	158.425	71.898
34	700	140	1083.555	73.677	226.733	73.923	1038.444	91.304	170.329	92.356
35	800	5	241.888	72.291	134.640	72.823	217.222	95.668	109.880	96.042
36	800	10	395.444	68.875	138.566	68.522	379.777	95.995	115.993	95.907
37	800	80	1125.666	81.188	190.364	79.345	985.111	115.559	208.808	117.084
38	900	5	354.222	96.293	203.823	97.181	309.333	126.913	144.920	126.650
39	900	10	474.777	88.159	174.992	87.775	428.333	121.855	176.755	122.011
40	900	90	2051.555	118.983	244.079	116.361	1612.888	149.598	200.122	149.907

Table 3.2: The results of various approaches on instances with 2 negative weights under model P1

			Objective function value			%Error	
Instance	n	p	best	ACO	ABC	ACO	ABC
1	100	5	5300	5300	5300	0.00	0.00
2	100	10	3724	3724	3724	0.00	0.00
3	100	10	3541	3541	3541	0.00	0.00
4	100	20	2450	2450	2450	0.00	0.00
5	100	33	878	878	878	0.00	0.00
6	200	5	7485	7485	7485	0.00	0.00
7	200	10	**5311**	5327	5311	0.30	**0.00**
8	200	20	4050	4050	4050	0.00	0.00
9	200	40	**2458**	2471	2458	0.53	**0.00**
10	200	67	986	986	986	0.00	0.00
11	300	5	7522	7522	7522	0.00	0.00
12	300	10	6494	6494	6494	0.00	0.00
13	300	30	4122	4122	4122	0.00	0.00
14	300	60	2679	2679	2679	0.00	0.00
15	300	100	1534	1534	1534	0.00	0.00
16	400	5	7994	7994	7994	0.00	0.00
17	400	10	6899	6899	6899	0.00	0.00
18	400	40	4569	4569	4569	0.00	0.00
19	400	80	2691	2691	2695	0.00	0.15
20	400	133	1615	1615	1616	0.00	0.06
21	500	5	9044	9044	9044	0.00	0.00
22	500	10	8444	8444	8444	0.00	0.00
23	500	50	4475	4475	4475	0.00	0.00
24	500	100	2805	2805	2810	0.00	0.18
25	500	167	1633	1633	1635	0.00	0.12
26	600	5	9803	9803	9803	0.00	0.00
27	600	10	8190	8190	8190	0.00	0.00
28	600	60	**4339**	4343	4339	0.09	**0.00**
29	600	120	2913	2913	2913	0.00	0.00
30	600	200	1840	1840	1842	0.00	0.11
31	700	5	10015	10015	10015	0.00	0.00
32	700	10	9211	9211	9211	0.00	0.00
33	700	70	4575	4575	4576	0.00	0.02
34	700	140	2830	2830	2830	0.00	0.00
35	800	5	10319	10319	10319	0.00	0.00
36	800	10	9862	9862	9862	0.00	0.00
37	800	80	4921	4921	4921	0.00	0.00
38	900	5	10993	10993	10993	0.00	0.00
39	900	10	9347	9347	9347	0.00	0.00
40	900	90	**5029**	5031	5029	0.04	**0.00**

Table 3.3: The results of various approaches on instances with 5 negative weights under model P1

			Objective function value			%Error	
Instance	n	p	best	GA	ABC	GA	ABC
1	100	5	4681	4730	4730	1.05	1.05
2	100	10	2915	2915	2916	0.00	0.03
3	100	10	2529	2529	2532	0.00	0.12
4	100	20	1432	1432	1464	0.00	2.23
5	100	33	61	61	61	0.00	0.00
6	200	5	7061	7061	7064	0.00	0.04
7	200	10	4812	4846	4858	0.71	0.96
8	200	20	3399	3399	3433	0.00	1.00
9	200	40	**1918**	1919	1918	0.05	**0.00**
10	200	67	514	514	514	0.00	0.00
11	300	5	7290	7290	7290	0.00	0.00
12	300	10	6201	6201	6201	0.00	0.00
13	300	30	3798	3798	3808	0.00	0.26
14	300	60	**2263**	2269	2263	0.27	**0.00**
15	300	100	**1151**	1153	1151	0.17	**0.00**
16	400	5	7787	7787	7787	0.00	0.00
17	400	10	6723	6735	6727	0.18	0.06
18	400	40	4219	4219	4221	0.00	0.05
19	400	80	2420	2420	2421	0.00	0.04
20	400	133	1346	1346	1349	0.00	0.22
21	500	5	8888	8888	8897	0.00	0.10
22	500	10	8230	8230	8230	0.00	0.00
23	500	50	4256	4256	4288	0.00	0.75
24	500	100	2538	2538	2542	0.00	0.16
25	500	167	1394	1394	1400	0.00	0.43
26	600	5	9655	9655	9655	0.00	0.00
27	600	10	8038	8038	8038	0.00	0.00
28	600	60	4043	4043	4055	0.00	0.30
29	600	120	2698	2698	2727	0.00	1.07
30	600	200	**1603**	1604	1603	0.06	**0.00**
31	700	5	9909	9909	9909	0.00	0.00
32	700	10	8935	8955	8955	0.22	0.22
33	700	70	4411	4411	4424	0.00	0.29
34	700	140	2605	2605	2628	0.00	0.88
35	800	5	10230	10230	10230	0.00	0.00
36	800	10	9735	9735	9742	0.00	0.07
37	800	80	4723	4723	4742	0.00	0.40
38	900	5	10860	10860	10860	0.00	0.00
39	900	10	9070	9070	9070	0.00	0.00
40	900	90	4862	4862	4881	0.00	0.39

Table 3.4: The results of various approaches on instances with 10 negative weights under model P1

				Objective function value			%Error		
Instance	n	p	best	GA	ACO	ABC	GA	ACO	ABC
1	100	5	3611	3611	3611	3611	0.00	0.00	0.00
2	100	10	1247	1247	1247	1247	0.00	0.00	0.00
3	100	10	1029	1029	1029	1029	0.00	0.00	0.00
4	100	20	-52	-52	-52	-52	0.00	0.00	0.00
5	100	33	-1143	-1143	-1143	-1143	0.00	0.00	0.00
6	200	5	6374	6374	6374	6374	0.00	0.00	0.00
7	200	10	4095	4095	4095	4095	0.00	0.00	0.00
8	200	20	2575	2575	2575	2575	0.00	0.00	0.00
9	200	40	**1085**	1088	1091	1085	0.28	0.55	**0.00**
10	200	67	-204	-204	-204	-204	0.00	0.00	0.00
11	300	5	6756	6756	6756	6756	0.00	0.00	0.00
12	300	10	5610	5610	5610	5610	0.00	0.00	0.00
13	300	30	3193	3193	3193	3193	0.00	0.00	0.00
14	300	60	1480	1480	1489	1480	0.00	0.61	0.00
15	300	100	632	635	632	632	0.47	0.00	0.00
16	400	5	7426	7426	7426	7426	0.00	0.00	0.00
17	400	10	6292	6292	6292	6292	0.00	0.00	0.00
18	400	40	3693	3693	3693	3693	0.00	0.00	0.00
19	400	80	2012	2013	2012	2013	0.05	0.00	0.05
20	400	133	910	910	910	911	0.00	0.00	0.11
21	500	5	8630	8630	8630	8630	0.00	0.00	0.00
22	500	10	7765	7765	7845	7765	0.00	1.03	0.00
23	500	50	3795	3795	3795	3795	0.00	0.00	0.00
24	500	100	2151	2151	2151	2153	0.00	0.00	0.09
25	500	167	990	990	990	994	0.00	0.00	0.40
26	600	5	9400	9400	9400	9400	0.00	0.00	0.00
27	600	10	7651	7651	7651	7651	0.00	0.00	0.00
28	600	60	3576	3576	3578	3577	0.00	0.06	0.03
29	600	120	2358	2359	2358	2360	0.04	0.00	0.08
30	600	200	1196	1199	1196	1196	0.25	0.00	0.00
31	700	5	9519	9688	9688	9688	1.78	1.78	1.78
32	700	10	8362	8418	8418	8418	0.67	0.67	0.67
33	700	70	**4142**	4144	4147	4142	0.05	0.12	**0.00**
34	700	140	2219	2219	2219	2220	0.00	0.00	0.05
35	800	5	10039	10039	10039	10039	0.00	0.00	0.00
36	800	10	9415	9415	9415	9415	0.00	0.00	0.00
37	800	80	4384	4384	4384	4385	0.00	0.00	0.02
38	900	5	10696	10696	10696	10696	0.00	0.00	0.00
39	900	10	8535	8535	8535	8535	0.00	0.00	0.00
40	900	90	4595	4595	4599	4596	0.00	0.09	0.02

Table 3.5: The results of various approaches on instances with half negative weights under model P1

				Objective function value			%Error		
Instance	n	p	best	GA	ACO	ABC	GA	ACO	ABC
1	100	5	-7651	-7651	-7651	-7651	0.00	0.00	0.00
2	100	10	-9445	-9445	-9445	-9445	0.00	0.00	0.00
3	100	10	-12398	-12398	-12398	-12398	0.00	0.00	0.00
4	100	20	-11507	-11507	-11507	-11507	0.00	0.00	0.00
5	100	33	**-10930**	-10811	-10811	-10930	1.09	1.09	**0.00**
6	200	5	-9971	-9971	-9971	-9971	0.00	0.00	0.00
7	200	10	-10403	-10403	-10403	-10403	0.00	0.00	0.00
8	200	20	-13912	-13912	-13901	-13912	0.00	0.08	0.00
9	200	40	-13997	-13997	-13997	-13997	0.00	0.00	0.00
10	200	67	-12437	-12437	-12437	-12437	0.00	0.00	0.00
11	300	5	-10271	-10271	-10271	-10271	0.00	0.00	0.00
12	300	10	-14850	-14850	-14850	-14850	0.00	0.00	0.00
13	300	30	-13557	-13557	-13557	-13557	0.00	0.00	0.00
14	300	60	-17676	-17676	-17676	-17676	0.00	0.00	0.00
15	300	100	-14437	-14437	-14437	-14437	0.00	0.00	0.00
16	400	5	-10792	-10792	-10792	-10792	0.00	0.00	0.00
17	400	10	-11583	-11583	-11583	-11583	0.00	0.00	0.00
18	400	40	-16286	-16286	-16286	-16286	0.00	0.00	0.00
19	400	80	-14200	-14200	-14200	-14199	0.00	0.00	0.01
20	400	133	-16362	-16362	-16361	-16362	0.00	0.01	0.00
21	500	5	-11296	-11296	-11296	-11296	0.00	0.00	0.00
22	500	10	-16588	-16588	-16588	-16588	0.00	0.00	0.00
23	500	50	-15272	-15272	-15272	-15271	0.00	0.00	0.01
24	500	100	**-17427**	-17221	-17221	-17427	1.18	1.18	**0.00**
25	500	167	-17924	-17924	-17922	-17923	0.00	0.01	0.01
26	600	5	-13060	-13060	-13060	-13060	0.00	0.00	0.00
27	600	10	-16204	-16204	-16179	-16204	0.00	0.15	0.00
28	600	60	-22970	-22970	-22970	-22970	0.00	0.00	0.00
29	600	120	-17796	-17796	-17796	-17796	0.00	0.00	0.00
30	600	200	-21333	-21333	-21333	-21332	0.00	0.00	0.00
31	700	5	-11466	-11396	-11396	-11396	0.61	0.61	0.61
32	700	10	-30465	-30465	-30456	-30465	0.00	0.03	0.00
33	700	70	-16917	-16914	-16917	-16917	0.02	0.00	0.00
34	700	140	**-24017**	-23803	-23805	-24017	0.89	0.88	**0.00**
35	800	5	-14709	-14709	-14709	-14709	0.00	0.00	0.00
36	800	10	-21934	-21934	-21934	-21934	0.00	0.00	0.00
37	800	80	-21038	-21038	-21036	-21036	0.00	0.01	0.01
38	900	5	-21059	-21059	-21059	-21059	0.00	0.00	0.00
39	900	10	-38980	-38980	-38980	-38980	0.00	0.00	0.00
40	900	90	-19350	-19350	-19350	-19347	0.00	0.00	0.02

Table 3.6: The results of various approaches on instances with 2 negative weights under model P2

			Objective function value			%Error	
Instance	n	p	best	ACO	ABC	ACO	ABC
1	100	5	5499	5499	5499	0.00	0.00
2	100	10	**4009**	4029	4009	0.50	**0.00**
3	100	10	3920	3920	3920	0.00	0.00
4	100	20	2845	2845	2845	0.00	0.00
5	100	33	1292	1292	1292	0.00	0.00
6	200	5	7590	7590	7590	0.00	0.00
7	200	10	**5457**	5471	5457	0.26	**0.00**
8	200	20	4281	4281	4281	0.00	0.00
9	200	40	**2702**	2713	2702	0.41	**0.00**
10	200	67	1213	1213	1214	0.00	0.08
11	300	5	7574	7574	7574	0.00	0.00
12	300	10	6584	6584	6584	0.00	0.00
13	300	30	4259	4259	4259	0.00	0.00
14	300	60	**2888**	2897	2888	0.31	**0.00**
15	300	100	1706	1706	1706	0.00	0.00
16	400	5	8034	8034	8034	0.00	0.00
17	400	10	**6943**	6945	6943	0.03	**0.00**
18	400	40	4713	4713	4713	0.00	0.00
19	400	80	2815	2815	2817	0.00	0.07
20	400	133	1747	1747	1749	0.00	0.11
21	500	5	9100	9100	9100	0.00	0.00
22	500	10	8487	8487	8487	0.00	0.00
23	500	50	4577	4577	4577	0.00	0.00
24	500	100	2923	2923	2927	0.00	0.14
25	500	167	1777	1777	1779	0.00	0.11
26	600	5	9827	9827	9827	0.00	0.00
27	600	10	8217	8217	8217	0.00	0.00
28	600	60	**4453**	4457	4453	0.09	**0.00**
29	600	120	3016	3016	3019	0.00	0.10
30	600	200	1973	1973	1976	0.00	0.15
31	700	5	10038	10038	10038	0.00	0.00
32	700	10	9251	9251	9251	0.00	0.00
33	700	70	4654	4654	4656	0.00	0.04
34	700	140	2956	2956	2956	0.00	0.00
35	800	5	10336	10336	10336	0.00	0.00
36	800	10	**9897**	9931	9897	0.34	**0.00**
37	800	80	5015	5015	5017	0.00	0.04
38	900	5	11014	11014	11014	0.00	0.00
39	900	10	9350	9377	9377	0.29	0.29
40	900	90	**5111**	5114	5111	0.06	**0.00**

Table 3.7: The results of various approaches on instances with 5 negative weights under model P2

			Objective function value			%Error	
Instance	n	p	best	GA	ABC	GA	ABC
1	100	5	5324	5324	5324	0.00	0.00
2	100	10	3696	3696	3696	0.00	0.00
3	100	10	3592	3592	3592	0.00	0.00
4	100	20	2460	2460	2460	0.00	0.00
5	100	33	994	994	994	0.00	0.00
6	200	5	7266	7266	7266	0.00	0.00
7	200	10	5224	5224	5224	0.00	0.00
8	200	20	4034	4034	4034	0.00	0.00
9	200	40	**2539**	2540	2539	0.04	**0.00**
10	200	67	**1074**	1077	1074	0.28	**0.00**
11	300	5	7440	7440	7440	0.00	0.00
12	300	10	6511	6511	6511	0.00	0.00
13	300	30	4187	4187	4187	0.00	0.00
14	300	60	**2773**	2785	2773	0.43	**0.00**
15	300	100	**1576**	1581	1576	0.32	**0.00**
16	400	5	7870	7870	7870	0.00	0.00
17	400	10	6849	6849	6849	0.00	0.00
18	400	40	4589	4589	4589	0.00	0.00
19	400	80	**2737**	2741	2737	0.15	**0.00**
20	400	133	1703	1703	1706	0.00	0.18
21	500	5	8980	8980	8980	0.00	0.00
22	500	10	8403	8403	8403	0.00	0.00
23	500	50	4520	4520	4520	0.00	0.00
24	500	100	2853	2853	2853	0.00	0.00
25	500	167	1735	1735	1738	0.00	0.17
26	600	5	9719	9719	9719	0.00	0.00
27	600	10	8137	8137	8137	0.00	0.00
28	600	60	**4384**	4385	4384	0.02	**0.00**
29	600	120	2965	2965	2965	0.00	0.00
30	600	200	1939	1939	1941	0.00	0.10
31	700	5	9915	9968	9968	0.53	0.53
32	700	10	9179	9179	9179	0.00	0.00
33	700	70	**4619**	4624	4619	0.11	**0.00**
34	700	140	**2910**	2913	2910	0.10	**0.00**
35	800	5	10286	10286	10286	0.00	0.00
36	800	10	9803	9803	9803	0.00	0.00
37	800	80	4967	4967	4970	0.00	0.06
38	900	5	10914	10914	10914	0.00	0.00
39	900	10	9305	9305	9305	0.00	0.00
40	900	90	5075	5075	5078	0.00	0.06

Table 3.8: The results of various approaches on instances with 10 negative weights under model P2

				Objective function value			%Error		
Instance	n	p	best	GA	ACO	ABC	GA	ACO	ABC
1	100	5	4826	4826	4826	4826	0.00	0.00	0.00
2	100	10	2976	2976	2976	2976	0.00	0.00	0.00
3	100	10	3341	3341	3341	3341	0.00	0.00	0.00
4	100	20	1854	1854	1854	1854	0.00	0.00	0.00
5	100	33	533	533	533	533	0.00	0.00	0.00
6	200	5	6787	6787	6787	6787	0.00	0.00	0.00
7	200	10	4949	4949	4949	4949	0.00	0.00	0.00
8	200	20	3812	3812	3812	3812	0.00	0.00	0.00
9	200	40	**2389**	2391	2392	2389	0.08	0.13	**0.00**
10	200	67	903	904	903	903	0.11	0.00	0.00
11	300	5	7136	7136	7136	7136	0.00	0.00	0.00
12	300	10	6395	6395	6395	6395	0.00	0.00	0.00
13	300	30	4011	4011	4011	4011	0.00	0.00	0.00
14	300	60	2564	2564	2564	2564	0.00	0.00	0.00
15	300	100	1457	1462	1457	1461	0.34	0.00	0.27
16	400	5	7624	7624	7624	7624	0.00	0.00	0.00
17	400	10	6668	6668	6668	6668	0.00	0.00	0.00
18	400	40	4437	4437	4437	4437	0.00	0.00	0.00
19	400	80	2629	2633	2629	2632	0.15	0.00	0.11
20	400	133	1621	1623	1621	1623	0.12	0.00	0.12
21	500	5	8800	8800	8800	8800	0.00	0.00	0.00
22	500	10	8291	8291	8291	8291	0.00	0.00	0.00
23	500	50	4337	4337	4337	4337	0.00	0.00	0.00
24	500	100	2748	2779	2779	2784	1.13	1.13	1.31
25	500	167	1636	1636	1636	1638	0.00	0.00	0.12
26	600	5	9528	9528	9528	9528	0.00	0.00	0.00
27	600	10	8004	8004	8004	8004	0.00	0.00	0.00
28	600	60	4296	4296	4296	4296	0.00	0.00	0.00
29	600	120	2896	2897	2896	2898	0.03	0.00	0.07
30	600	200	1850	1850	1851	1855	0.00	0.05	0.27
31	700	5	9820	9820	9820	9820	0.00	0.00	0.00
32	700	10	9053	9053	9053	9053	0.00	0.00	0.00
33	700	70	4566	4572	4566	4568	0.13	0.00	0.04
34	700	140	2833	2835	2833	2835	0.07	0.00	0.07
35	800	5	10142	10142	10142	10142	0.00	0.00	0.00
36	800	10	9637	9637	9637	9637	0.00	0.00	0.00
37	800	80	4875	4875	4878	4877	0.00	0.06	0.04
38	900	5	10800	10800	10839	10800	0.00	0.36	0.00
39	900	10	9201	9201	9201	9201	0.00	0.00	0.00
40	900	90	5026	5026	5028	5026	0.00	0.04	0.00

Table 3.9: The results of various approaches on instances with half negative weights under model P2

				Objective function value			%Error		
Instance	n	p	best	GA	ACO	ABC	GA	ACO	ABC
1	100	5	-635	-635	-635	-635	0.00	0.00	0.00
2	100	10	-1245	-1245	-1245	-1245	0.00	0.00	0.00
3	100	10	-1131	-1131	-1131	-1131	0.00	0.00	0.00
4	100	20	-1477	-1477	-1477	-1477	0.00	0.00	0.00
5	100	33	-1687	-1687	-1687	-1687	0.00	0.00	0.00
6	200	5	-1163	-1163	-1163	-1163	0.00	0.00	0.00
7	200	10	-1360	-1360	-1360	-1360	0.00	0.00	0.00
8	200	20	-1765	-1765	-1764	-1765	0.00	0.06	0.00
9	200	40	-2212	-2212	-2212	-2212	0.00	0.00	0.00
10	200	67	-1815	-1815	-1815	-1815	0.00	0.00	0.00
11	300	5	-797	-797	-797	-797	0.00	0.00	0.00
12	300	10	-1290	-1290	-1290	-1290	0.00	0.00	0.00
13	300	30	-1709	-1709	-1697	-1708	0.00	0.70	0.06
14	300	60	-2224	-2224	-2219	-2223	0.00	0.22	0.04
15	300	100	**-2154**	-2152	-2151	-2154	0.09	0.14	**0.00**
16	400	5	-932	-932	-932	-932	0.00	0.00	0.00
17	400	10	-1318	-1254	-1254	-1233	4.86	4.86	6.45
18	400	40	-2096	-2096	-2096	-2089	0.00	0.00	0.33
19	400	80	-2119	-2119	-2119	-2119	0.00	0.00	0.00
20	400	133	**-2295**	-2291	-2290	-2295	0.17	0.22	**0.00**
21	500	5	-687	-687	-622	-687	0.00	9.46	0.00
22	500	10	-1111	-1111	-1098	-1098	0.00	1.17	1.17
23	500	50	-1933	-1933	-1915	-1933	0.00	0.93	0.00
24	500	100	-2221	-2216	-2221	-2221	0.23	0.00	0.00
25	500	167	**-2379**	-2376	-2368	-2379	0.13	0.46	**0.00**
26	600	5	-820	-820	-820	-820	0.00	0.00	0.00
27	600	10	-1053	-1053	-1053	-1053	0.00	0.00	0.00
28	600	60	-2119	-2119	-2107	-2117	0.00	0.57	0.09
29	600	120	-2198	-2198	-2197	-2197	0.00	0.05	0.05
30	600	200	**-2321**	-2320	-2320	-2321	0.04	0.04	**0.00**
31	700	5	-748	-748	-748	-748	0.00	0.00	0.00
32	700	10	-1030	-1030	-975	-1030	0.00	5.34	0.00
33	700	70	-2009	-2009	-2009	-2005	0.00	0.00	0.20
34	700	140	**-2437**	-2436	-2427	-2437	0.04	0.41	**0.00**
35	800	5	-855	-855	-855	-855	0.00	0.00	0.00
36	800	10	-985	-985	-985	-985	0.00	0.00	0.00
37	800	80	-2278	-2278	-2263	-2276	0.00	0.66	0.09
38	900	5	-607	-607	-522	-607	0.00	14.0	0.00
39	900	10	-901	-901	-901	-901	0.00	0.00	0.00
40	900	90	-2347	-2347	-2343	-2347	0.00	0.17	0.00

Table 3.10: Summary table

		GA		ACO		ABC		
Model	Weights	W	TE	W	TE	W	TE	BKV-I
P1	2-neg	-	-	4	0.96	6	0.64	4
P1	5-neg	8	2.71	-	-	24	11.12	4
P1	10-neg	8	3.59	8	4.91	11	3.30	2
P1	half-neg	5	3.79	10	4.05	6	0.67	3
P2	2-neg	-	-	9	2.29	10	1.13	8
P2	5-neg	9	1.98	-	-	6	1.10	8
P2	10-neg	9	2.16	6	1.77	10	2.42	1
P2	half-neg	7	5.56	18	39.46	9	8.48	5

where first 5 vertices have negative weights was considered in [109] only, whereas the case where first 2 vertices have negative weights was considered in [108] only. All other cases were considered by both the papers. Hence, the results for GA and ACO are not available for instances with 2 negative weights and 5 negative weights respectively. Like GA and ACO approaches, we have executed our hybrid ABC approach 5 times on each instance and reported the average results.

For both the models, i.e., P1 and P2, we have reported the total CPU time that is averaged over all the instances that are derived from the same OR-Library instance. This is done to ensure conformity with [109] and [108]. Table 3.1 reports these times and the next paragraph further explains how these times have been computed. Table 3.2 to Table 3.9 present the results of ABC algorithm on various types of instances and compare them with those of GA and ACO. Table 3.2, Table 3.3, Table 3.4 and Table 3.5 report the results of various approaches for model P1 on instances with 2 negative weights, 5 negative weights, 10 negative weights and half negative weights respectively, whereas Table 3.6, Table 3.7, Table 3.8 and Table 3.9 does the same for model P2. As mentioned already, performances of GA and ACO were not evaluated on instances with 2 negative weights and 5 negative weights respectively, and hence, tables Table 3.2 and Table 3.6 report the results of ABC and ACO only, whereas Table 3.3 and Table 3.7 report the results of ABC and GA only. Results for GA and ACO are taken from their respective papers. The columns under the common heading *Objective function value* report the objective function value averaged over 5 runs for various approaches,

whereas columns under the common heading *%Error* report the relative error of various approaches on each instance. The relative error is defined as follows:

$$\frac{f - f_{O/B}}{|f_{O/B}|} \times 100$$

where f is the objective function value obtained by the algorithm and $f_{O/B}$ is the optimal or the best known value so far obtained. For some instances, our ABC approach has found a value better than the best known value. In such cases, we have replaced the best known value with new best known value found by our ABC algorithm. Such cases are reported in bold font in these tables.

As mentioned in the previous paragraph, Table 3.1 reports the average total CPU times in seconds of GA, ACO and ABC approaches for each of the two models (P1 and P2). The first three columns represent the problem number of the original OR-Library instance, number of nodes and number of facilities. columns 4-7 report the average time for model P1 and columns 8-11 report the average time for the model P2. As explained earlier, performances of GA and ACO were not evaluated on instances with 2 negative weights and 5 negative weights respectively. Hence, to ensure the fair comparison, we have computed average total CPU times for our approach in two ways. For comparison with the GA, the averages are computed using instances with 5 negative weights, 10 negative weights and half negative weights, whereas for comparison with ACO, the averages are computed using instances with 2 negative weights, 10 negative weights, and half negative weights. Hence, we have two columns labelled ABC for each of the two models. Columns 5 and 9 report the time of ABC approach for comparison with GA, whereas columns 7 and 11 report the time of ABC approach for comparison with ACO. Again, the data for GA and ACO are taken from their respective papers.

Table 3.10 summarizes the results. This table reports for each approach on each instance group of 40, the number of instances for which the approach in question found result inferior to best known value (column W) and sumtotal of relative error (column TE). This table also reports the number of instances in each instance group where our ABC approach found the new best known value (column $BKV\text{-}I$).

From these tables, some interesting observation can be made. Results of different approaches vary according to the types of instances. ABC approach improves the best known values for more than 10% of the instances (35 out of 320). Most of these instances are those with a relatively large value of p. Barring few exceptions, there is not much

difference in the performance of various approaches on the instances with small values of p. Only when the value of p is large, the performance of different approaches tends to differ significantly. However, none of the approaches can be considered as clearly superior to others on all types of instances with large value of p. A justification for the difficulty of various approaches on instances with large value of p has been provided already in Section 2.4.1 of Chapter 2.

We can observe that our ABC approach performs better than ACO on 5 out of 6 instance groups in terms of total error. Our approach is better on 2 out of 6 instance groups if we compare two approaches in terms of the number of instances where an approach fails to reach the best known value. This shows that whenever ABC approach fails to reach the best known value, its solution is closer to best known value in comparison to the solution of ACO under the similar situation. Overall, there are 55 and 52 instances (out of 240) where ACO and ABC fail to reach the best known value. ACO approach performs much worse in comparison to our ABC approach on instances with half negative weights under both the models.

On the other hand, GA, in general, fares better than ABC. There are 3 instance groups where ABC performs better than GA in terms of total error, whereas on other 3 groups GA performs better. There is only one instance group where GA fails to reach the best known value on a higher number of instances than ABC. However, GA performs much worse in comparison to ABC in terms of total error on instances with half negative weights under model P1.

As far as execution times of various approaches are concerned, GA and ACO approaches were executed on a 1.7GHz Pentium 4 system which is different from the system used to execute our ABC approach. Therefore, execution times can not be compared precisely. However, a rough comparison can always be made. Even after compensating for differences in processing speed, we can safely say that our approach is faster than GA on most of the instances. However, ACO is faster than our approach.

3.4 Conclusions

In this Chapter, we have proposed an ABC algorithm based approach for solving the p-median problem with positive and negative weights. We have compared the results of our approach with two best approaches, viz. GA and ACO on the standard benchmark

instances of the problem. Comparison with ACO approach is specially significant as both ABC and ACO are swarm intelligence based approaches. ABC approach is able to improve the best known values for slightly more than 10% of the instances. Though the relative performance of different approaches varies according to the types of instances, the overall performance of GA is clearly better than ABC approach in terms of solution quality, but ABC approach is faster. On the other hand, ABC approach is much better than ACO approach on instances where half of the weights are negative under both the models, but at the expense of larger execution times.

Chapter 4

Cooperative maximum coverage location problem

4.1 Introduction

The cooperative maximum covering location problem (CMCLP) was introduced by Berman et al. [31]. In this problem, it is assumed that each facility generates a signal whose strength decreases over distance, according to some signal strength function. A demand point receives the signals from all the facilities, and it is considered as covered if the strength of the combined signal received by it from all the facilities exceeds a certain threshold. Hence, the CMCLP assumes cooperative coverage model, where all the facilities cooperate in providing the coverage. This coverage model differs from the one used in classical maximum covering location problem (MCLP) introduced by Church and Revelle [2], where the signal received from the single closest facility determines whether a demand point is covered or not, thus it is referred as individual coverage model.

The individual coverage model may not always be appropriate as it may result in poor quality of solutions by underestimating the coverage provided by the group of facilities together; which results in deployment of more number of resources than actually needed.

The signal can be physical or non-physical. A physical signal such as radio signal propagates along the straight line path between two points, whereas a non-physical signal may not. An example of a non-physical signal is the service time of an ambulance

to reach a particular point from its rest location. Rather than travelling on a straight line path, an ambulance has to make use of the road network to reach the desired point from its rest position. Depending upon the traffic on the road and other factors, service time can vary.

The applications of the cooperative cover models are discussed by Berman et al. in [31], in which they discussed the applications that belong to both physical signal class and non-physical signal class of cover problems. For example; applications such as installation of light towers to provide lighting to an area, location of warning sirens and location of cell phone towers belong to the physical signal class. The main aspect of the physical signal class of cooperative cover models is - the signal received by a particular demand point is the combined signal from all the facilities. In the case of location of cell towers the demand point (mobile phone user) is considered as covered if the probability of establishing a successful connection exceeds a certain threshold value. On the other hand, applications such as an emergency response system design, belong to the non-physical signal class. In such systems having a single system within the coverage of demand point is not sufficient, to ensure the coverage all the time because it may become busy at some point. To deal with these type of issues, the demand point must be within the coverage radius of several facilities.

The potential application for cooperative cover problems include situations, where the facilities are generally unreliable and the customer needs to search for the facility until the service is done. In such situations, the customer is considered as covered; if the expected total travelling cost is below the threshold value, until the service is done. Another application of cooperative cover includes models with the assumption of partial cover by the facilities, where a single facility is not sufficient to provide the coverage.

One familiar example for cooperative maximum covering location problem, on a network is 'Pizza Delivery System '. Consider a set of pizza centres promoting their service as 'Delivery In Only 30 Minutes ', Supposing a typical location; the walk-in traffic time is distributed as a normal variable, with the mean of 20 minutes and standard deviation of 2 minutes respectively, the trading area is defined as the customers who could be covered within 30 minutes, at least 75% of the allocated time. For an individual coverage model, the customers should be within the coverage of $30 - 20 - 0.7 \times 2 = 8.6$ minutes ($0.7 = 75\%$ of the standard normal distribution) to get the

delivery within 30 minutes. For instance, a customer would be located at a distance of 15 minutes away from the pizza center and has only 50% chances to get the pizza delivery in time. And this customer has another pizza center which is at a distance of 9 minutes away, and has only 69% chances to reach; both centers are far from the coverage area. Assuming the dispatcher redirects the order to the center which is active at the moment(least busy), and the order preparation times are independent at both the centers, Here in this particular situation, it is easy to deliver the order to the customer within 30 minutes with the probability of $1-(1-0.5)(1-0.69) = 0.84$ which equals to 84%. With this example it is proved that the customer has the chances of being covered by multiple locations, by providing cooperative coverage.

In the literature, covering problems are analyzed for plane [110] [111] [112] as well as for network topology [113] [37] assuming the discrete demand. Covering problems are also analyzed by assuming the continuously distributed demand in [114] [115]. The planar version of the CMCLP is solved using optimal and heuristic algorithms by Berman et al. in [31]. In [31] the authors proposed an algorithm which is based on the big triangle and small triangle approach [116] for the $2-facility$ problem with the assumption that one of the facilities location is known. For more than $2-facilities$ problem they proposed a heuristic method. The discrete version of the CMCLP was considered by Berman et al. in [36], where the location space is restricted to the nodes of the network. The discrete version allowing the facilities to be located at both the nodes and along the edges is solved by Averbakh et al. in [35]. In which the authors proposed greedy methods and local search based heuristics to solve the $p-facility$ problem.

In this chapter, we have proposed an artificial bee colony algorithm based approach for solving the CMCLP for a network version of $p-facility$ problem where facilities can be located both at the nodes and along the edges. We have compared our results with two interchange heuristic methods proposed in [35]. In comparison to these methods, our approach obtained better quality results on most of the instances.

The remainder of this chapter is organized as follows: Section 4.2 formally defines CMCLP problem and introduces the notational conventions used in this chapter. Section 4.3 describes our approach to solve the CMCLP. Section 4.4 reports the computational results and provides a comparative analysis of our algorithm. Finally, Section 4.5 outlines some concluding remarks.

Table 4.1: Summary of notations

n	The number of nodes or demand points, i.e., $n = \|V\|$
w_i	The weight associated with demand point $i \in V$
l_k	The length associated with each egde $e_k \in E$
X	The location space
p	The number of facilities to locate
$d_i(x_j)$	The distance between demand point $i \in N$ and facility j
$\phi(d)$	The strength of the signal at distance d from the facility, $\phi(d) = \max\{0, 1 - d/U\}$
$\Phi_i(x)$	The overall signal at point $i \in N$
U	The fraction of diameter of the network
T	The threshold for coverage

4.2 Formal problem definition

The CMCLP can be stated as follows:
Let $G = (V, E)$ be an undirected network, with $V = (1, ..., n)$ be the set of demand points and $E = (e_1, ..., en)$ be the set of edges connecting various demand points. Each demand point i has a non-negative real weight w_i indicating the total demand at this point. Each edge e_k has a positive length l_k. p facilities need to be located on the edges (including end demand points) to cover these demand points. A demand point is said to be covered if the combined strength of the signal received by it from all the facilities is not less than a threshold T. The CMCLP seeks a location vector X_p for these p facilities so as to maximize the sumtotal of the weights of the covered demand points.

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

The signal strength function is defined as the sum of all signals received by a demand point $i \in V$ from p facilities.

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

Location x of a facility along an edge e_k joining demand point j_1 with j_2 ($e_k = (j_1, j_2)$) is represented by an ordered pair (l_k, t) where t is the relative distance of x from j_1 with respect to l_k, i.e., $0 \leq t \leq 1$. The distance from a node $i \in V$ to this facility is defined as follows:

$$d\left(i,x\right)=\min\left\{d\left(i,j_1\right)+t\times l_k, d\left(i,j_2\right)+(1-t)\times l_k\right\}$$

Where $d(i,j_1)$ and $d(i,j_2)$ are the lengths of the shortest paths connecting node i to node j_1 and node j_2 respectively.

The notations used above are summarized in Table 4.1

To illustrate CMCLP, consider the network depicted in Figure 4.1. Let P=3, T=0.8, and the signal strength function be defined as below:

$$\phi(d)=\max\left\{0,1-\tfrac{d}{10}\right\}.$$

One possible solution that we can obtain by locating the facilities along the edges as well as on nodes is $X=\{(4),([7,8],0.6),([1,2],0.4)\}$, where the first point p_1 is located on node 4, the second point p_2 is located on edge $[7,8]$ at a relative distance of $t=0.6$ from node 7, and the last point p_3 is located on edge $[1,2]$ at a relative distance of $t=0.4$ from node 1. This solution produces an objective value of 29 covering all nodes.

In the above example, we have selected the points randomly. There might be a possibility of some other possible solutions, existing for the same network.

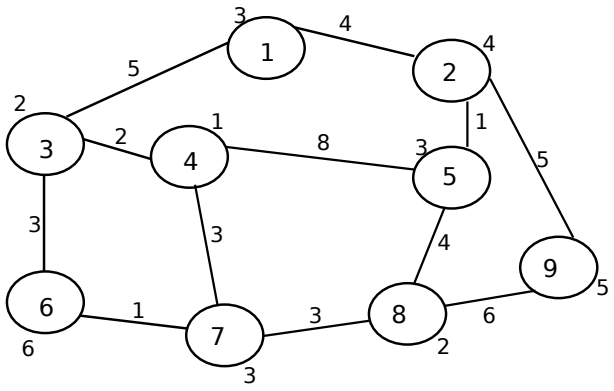


Figure 4.1: Example illustrating CMCLP

4.3 ABC approach for the CMCLP

This section presents salient features of our ABC approach for the CMCLP.

4.3.1 Fitness of a solution

We follow the same two level approach as used in [35] for determining the fitness of a solution. We say that a solution X' is better than another solution X, if solution X' either has a larger objective function value,

$$f(X', T) > f(X, T)$$

or, if solution X and solution X' have same objective function value, but solution X' provides larger total coverage.

$$\sum_{i \in V} \Phi_i(X') > \sum_{i \in V} \Phi_i(X)$$

4.3.2 Probability of selecting a food source

Like the approach described in the previous chapters, We have used the binary tournament selection method for selecting a food source for an onlooker. The better candidate solution is selected with the probability p_{onl} and the probability of selection of the worst candidate solution is $1 - p_{onl}$.

4.3.3 Initial solution

To generate an initial solution we have followed a method which is a mix of greediness and randomness. In this method, one facility at a time will be added to the current partial solution starting with an empty solution. In each iteration, to add a facility to the current partial solution S, we compute the set Y of all points $x \in G$ (including nodes $\in V$) that provide exact or greater coverage for yet uncovered nodes V'. The x points are considered on edges at equal intervals separated by a relative distance of 0.1. The condition of exact or greater coverage is based on the signal strength function defined. With the choice of the signal strength function used, it may happen during the iterations of the greedy method that there may not exist even one point which can provide exact coverage (specially in case of large instances). So we have chosen points providing either exact or greater coverage. Next, we evaluate the points based on how much coverage they are providing, then choose R best points from the set Y, and select one point randomly from R.

$$Y = V \cup \{x \in G | \phi_i(x) \geq T_i(X)\} \text{ for some } i \in V'$$

After locating a facility, the threshold is updated as follows:

$$T_i(S) = max\{0, T - \Phi_i((S)\} \text{ for } i \in V$$

This process is repeated till the desired number of facilities are located.

4.3.4 Neighboring solution generation

To generate a solution X' in the neighborhood of solution X, we delete F facilities from X randomly. Then, F facilities are added one-by-one from the set F^{new} containing all points $x \in G$ (including nodes $\in V$) which provide exact coverage for at least one yet uncovered node in V'. A formal definition of F^{new} is given below.

$$F^{new} = Z - X$$

where

$$Z = V \cup \{x \in G | \phi_i(x) = T_i(X)\} \text{ for some } i \in V'$$

To add each facility, we first choose R best points from F^{new} and then choose one randomly from R. Once a facility has been added, F^{new} is updated. The manner in which facilities are added is similar to Greedy 1 heuristic of [35] except for the fact that Greedy 1 always choose the best point, whereas we choose randomly a point from among R best points.

4.3.5 Other features

If an employed bee solution does not improve for a specified number of iterations say *limit*, then the associated employed bee becomes a scout. There is no restriction on the number of scout bees in an iteration. The number of scouts in a particular iteration depends on the number of employed bee solutions which have not improved for an exactly *limit* number of iterations. The scout bee is again made employed by assigning it to a new solution which is generated in the same manner as one of our initial solutions.

Algorithm 6: Pseudo-code of our hybrid ABC algorithm

```
Randomly generated solutions e_1, e2, ..., e_n ;
best_sol:= best among e_1, e2, ..., e_n;
while termination condition is not satisfied do
    for i = 1 to e_n do
        e' := Generate_Neighbor(e_i);
        if e' is better than e_i then
            e_i:=e'
        if e' is better than best_sol then
            best_sol:=e'
    for i = 1 to o_n do
        k_i := Binary_Tournament(e_1, e2, ..., e_n );
        onl_i := Generate_Neighbor(e_{k_i});
        if onl_i is better than best_sol then
            best_sol:=onl_i
    for i = 1 to o_n do
        if onl_i is better than e_{k_i} then
            e_{k_i}:=onl_i
    for i = 1 to e_n do
        if e_i is not improved over limit iterations then
            replace e_i with a random solution;
Best_sol := Local_Search(best_sol);
return Best_sol ;
```

4.3.6 Local search

We have used a local search on the best solution obtained through ABC algorithm in a bid to improve it further. In this local search, each facility x in a solution S is considered one-by-one and the best point b_x to relocate it is determined. For this, we compute the set F^{new} for solution $(S \setminus \{x\})$ and find the best point b_x to locate the next facility. If the solution $(S \setminus \{x\}) \cup \{b_x\})$ is better than S then we replace S with this new solution. This process is repeated till each facility is considered once.

Algorithm 6 provides the pseudo-code for our hybrid ABC approach, where n_e and n_o are respectively the number of employed bees and number of onlooker bees. *Gen-*

erate_Neighbor(e_i) and *Binary_Tournament(e_1 , e_2,..., e_n)* and *Local_Search(best_sol)* are the three functions. The first function generates a solution in the neighborhood of solution e_i as described in Section 4.3.4. The second function selects a solution for an onlooker bee from all employed bee solutions e_1, e_2,..., e_n as per Section 4.3.2 and the last function returns a solution obtained after applying local search on best solution as explained in Section 4.3.6.

4.4 Computational results

To test our ABC approach, we have used the same instances as used in [35]. These instances are generated randomly in the following manner; For each combination of (n, dgr) where n is the number of nodes and dgr is the average degree of nodes, five instances were generated by the Averbakh et al. [35]. In these instances $n \in \{40, 60, 80, 100, 120, 140, 160, 180, 200\}$ and $dgr \in \{5, 6, 7\}$. Like in [35], we have done all computational experiments by taking $p = 3, 4, 5$ for $n = 40, 60, 80$, $p = 4, 5, 6$ for $n = 100, 120, 140$ and $p = 5, 6, 7$ for $n = 160, 180, 200$. Three different values have been used for signal threshold values T, i.e., $T \in \{0.6, 0.8, 1.0\}$ and linear signal strength function $\phi(d) = \max\{0, 1 - d/U\}$ has been used like in [35]. The parameter U was determined in [35] as a fraction of the diameter of the network and $U_{\%} = 0.15, 0.25, 0.35$ for $T = 0.6, 0.8$ and $U_{\%} = 0.2, 0.3, 0.4$ for $T = 1.0$ were used.

Our ABC approach has been implemented in C and executed on a Linux based Intel Core i5 2400 system with 4 GB memory running at 3.10 GHz. In all our computational experiments, the number of employed bees (n_e) is taken to be 10, the number of onlooker bees (n_o) is taken to be 20, *limit* is set to 50, p_{onl} is set to 0.9 for $U_{\%} = 0.15, 0.25, 0.35$ for $T = 0.6$ and it is set 0.8 for remaining combinations of U and T, and $R = 20$. We have taken $F = 2$ for $n = 40, 60, 80$ and $p = 3, 4, 5$, $F = 3$ for $n = 100, 120, 140$ and $p = 4, 5, 6$, $F = 4$ for $n = 160, 180, 200$ and $p = 5, 6, 7$. Our ABC approach terminates after 500 iterations. All these parameter values were chosen empirically after a large number of trials.

We have compared the results of our ABC approach with interchange heuristics I1 and I2 proposed in [35]. Results are reported in the same format as used in [35]. Authors of [35] reported the average percentage improvement in solution quality by I2 with respect to I1 over all the instances with same T and $U_{\%}$. However, average execution

Table 4.2: Computational results

T	$U_{\%}$	%(ABC, I1)	%(ABC, I2)	R(I1)	R(I2)	R(ABC)
0.6	0.15	1.28	1.21	42.36	25.28	351.62
0.6	0.25	12.84	12.76	99.99	67.21	253.26
0.6	0.35	1.1	1.07	80.61	104.49	244.21
0.8	0.15	4.88	4.71	39.07	28.93	366.45
0.8	0.25	22.36	21.98	84.42	78.00	271.49
0.8	0.35	5.34	5.06	118.80	141.13	257.82
1.0	0.2	35.98	35.52	36.26	29.48	263.43
1.0	0.3	25.15	24.86	122.42	108.10	278.56
1.0	0.4	12.08	12.09	82.56	153.21	206.29

times are reported over all instances with $n = 200$ only, i.e., all instances having the maximum number of nodes. The percentage improvement in solution quality by method A over method B on a particular instance is $100 \times \frac{\phi_i(S_A) - \phi_i(S_B)}{\phi_i(S_B)}$ where S_A and S_B are the solutions obtained by methods A and B respectively on the particular instance under consideration. Table 4.2 reports the results of I1, I2, and our ABC approach. Data for I1 and I2 have been obtained from the corresponding author of [35] through personal communication. The first column in Table 4.2 represents the thresholds (T value), second column represents $U_{\%}$ values, third column (%(ABC,I1)) reports the average percentage improvement of our ABC approach over I1, fourth column (%(ABC,I2)) reports the average percentage improvement of our ABC approach over I2. And the last 3 columns (R(.)) report the average execution times of various approaches on the largest instance ($n = 200$).

The results show that the performance of our ABC algorithm is better compared to the Interchange heuristics. With the choice of our parameters, we are able to provide larger coverage than the already existing methods.

4.5 Conclusions

In this chapter, we have proposed an ABC algorithm based approach for the CMCLP and compared it with two interchange based heuristic methods proposed in the literature. Our ABC algorithm outperformed these two methods in terms of solution quality. However, it is slower than these methods. Our ABC algorithm based approach

is the first metaheuristic approach for the CMCLP. Averbakh et al. [35] also experimented with tabu search and variable neighborhood search, but results obtained by these approaches were only slightly better than those obtained by interchange heuristics and hence they did not present these metaheuristics. Therefore, population based metaheuristics seem more appropriate for this problem.

Chapter 5

p-center problem

5.1 Introduction

Given an undirected weighted graph $G = (V, E)$ where V denotes the set of vertices, E denotes the set of edges and $|V| = n$, the p-center problem seeks on this graph a set $Y \subset V$ of p vertices so that the maximum distance from any vertex in V and its nearest vertex in Y is minimized. The vertices of the graph can be considered as demand points and the vertices in Y as the location of facilities. The goal of p-center problem is to choose the locations of p facilities to serve n demand points so that the maximum distance of any demand point from its nearest facility becomes as small as possible. The vertices in set Y are called centers. We have used the terms centers and facility locations interchangeably throughout this chapter. The p-median problem considered in Chapter 2 is concerned with minimizing the average distance of demand points from their nearest facilities as minimizing the total distance is equivalent to minimizing the average distance. However, in several real-world applications, particularly those dealing with the location of emergency facilities such as fire stations, police stations, and ambulance depots, it is more appropriate to locate the facilities in such a manner so that the farthest travel distance/time between a demand point and its nearest facility is minimized. p-center problem is used to model such applications and has attracted increasing attention in recent years.

The p-center problem is proven to be $\mathcal{NP}$-hard in the literature by Kariv and Hakimi [117]. Chen and Chen [118] solved the p-center problem using a new relaxation algorithm. Exact algorithms have been proposed for some special cases of p-center problem

in [119] and [120]. Mladenović et al. [121] presented two tabu search heuristics and a variable neighborhood search heuristic for the p-center problem. Caruso et al. [122] developed another heuristic approach called Dominant which solves a series of set-covering problems according to a pre-defined maximum distance to get a solution to the p-center problem. Pacheco and Casado [123] proposed a new approach based on scatter search. Davidović et al.[124] proposed an approach called BCOi based on bee colony optimization (BCO) with improvement concept. Till date, BCOi is the best approach for the p-center problem. Several other heuristics and metaheuristics have also been proposed in the literature, e.g. [125], [126] and [127].

In this chapter, we have proposed two new swarm intelligence based approaches for the p-center problem. Our first approach is based on artificial bee colony (ABC) algorithm, whereas the latter approach is based on invasive weed optimization (IWO) algorithm. We have compared our approaches with BCOi, which is the best approach available in the literature, on the standard benchmark instances for the problem. Computational results show the effectiveness of proposed approaches in finding high quality solutions.

Both ABC and BCO algorithms are inspired by the foraging behavior of honey bee swarm. The difference between these algorithms lies in the simple rules for modelling the nectar collection process [128]. The differences are described as follows;

- The bees will have designated roles, viz. scouts, employed, and onlookers in the ABC algorithm, while in BCO all bees will perform the same algorithm steps.

- As mentioned already, each cycle of the ABC algorithm consists of employed bee phase and onlooker bee phase. In addition, at the end of every cycle, the scout bees are determined. A new solution is determined for each such scout bee to turn it back into an employed bee.

  There are 2 passes in each iteration of BCO algorithm, i.e., forward pass and backward pass. In forward pass, every bee functions like an employed bee in employed bee phase of ABC Algorithm. During the backward pass, all bees are divided into two groups – recruiters (R), and uncommitted bees ($B - R$, where B is the total number of bees in the population). The values of R and $B - R$ will vary from one backward pass to another. Recruiters are determined using some probability based selection method where bees associated with good quality

solutions have more chances of becoming a recruiter. Those bees which can not become recruiters abandon their solutions and become uncommitted bees. In the backward pass, the uncommitted bees choose a recruiter and generates a solution in the neighborhood of the solution of its recruiter like the onlooker bees in ABC Algorithm. However, unlike the onlooker bees, uncommitted bees retain the solution it generated for the forward pass of the next iteration.

- In the ABC algorithm, while onlookers and employed bees carry out the exploitation in the search space, the scouts are responsible for the exploration. BCO algorithm does not have any equivalent of scout bees. However, population is initialized every NC iterations in BCO algorithm.

The rest of this Chapter is organized as follows: Section 5.2 provides a formal definition of the p-center problem. Section 5.3 describes our ABC approach to the p-center problem. Section 5.4 describes our IWO approach to the p-center problem. Computational results are presented in Section 5.5. Finally, Section 5.6 provide some conclusions.

5.2 Formal description

The p-center problem can be defined formally as follows: Let $G = (V, E)$ be an undirected graph with vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and edge set E. The length of shortest path or distance from vertex v_i to vertex v_j is denoted as $d(v_i, v_j)$. The problem is to choose a set Y containing p vertices of G, in such a way that the maximum distance of a vertex in V to its closest vertex in Y is minimized, i.e., the solution is a subset $Y = \{y_1, y_2, \ldots, y_p\}$ of V that minimizes

$$\max_{i \in \{1,\ldots,n\}} \left\{ \min_{j \in \{1,\ldots,p\}} d(v_i, y_j) \right\} \tag{5.1}$$

Figure 5.1 illustrates a typical solution of p-center problem with $p = 3$ facilities and $n = 15$ demand points. In this figure, each demand point is shown by a circle (black or white). The demand points which are also the centers or the location of facilities

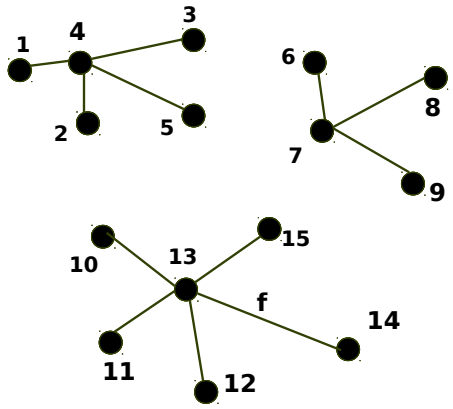


Figure 5.1: Example illustrating p-center problem

are shown by white circles, whereas the remaining demand points are shown by black circles. The demand points are connected to their closest centers, and this connection is shown by straight lines. The objective function value for this solution corresponds to distance between node 14 and center 13 which is the maximum distance over all the distances between a demand point and its closest center, i.e., $f = d(14, 13)$.

5.3 ABC approach for the p-center problem

This section describes our ABC approach for the p-center problem. The salient features of our ABC approach are discussed below in subsequent subsections.

5.3.1 Solution representation and fitness

A solution is represented by the subset of p vertices used for locating the facilities. The demand points are always assigned to the nearest facility present in the subset. For example, the solution in Figure 5.1 is represented as $Y = \{4, 7, 13\}$. The objective function is used as the fitness function, i.e., the fitness function value is calculated using the Equation (5.1). As p-center is a minimization problem, a more fit solution has a lesser fitness function value.

5.3.2 Food source selection for onlooker bees

For selecting a food source for an onlooker bee, binary tournament selection method is used where the candidate with better fitness is selected with probability p_{onl}.

5.3.3 Initial solution

The initial solutions for our ABC algorithm are generated in a purely random fashion. One location for a facility is selected at a time randomly from the not yet selected vertices of V and this process is repeated until p facilities are located.

5.3.4 Pre-processing

Like the BCOi approach of [124], the input data is pre-processed before starting our ABC approach with the intention of making some computations faster. This pre-processing involves two phases, and we have followed the notational conventions used in [124] to describe them. These two phases are described below

- In the first phase, all pair shortest distance matrix D is computed using Floyd's algorithm. Then from this distance matrix D, two new matrices are computed. Every row of D together with corresponding indices are sorted in increasing order, thereby creating two new matrices D_{sort} and D_{circle}. The matrix D_{sort} contains each row of D in sorted order, whereas each row of the matrix $D_{circles}$ provides the information about the first, second,...,n^{th} closest node to the node corresponding to the row. For the sake of illustration, consider $n = 4$ and

$$D = \begin{bmatrix} 0 & 3 & 4 & 1 \\ 3 & 0 & 6 & 8 \\ 4 & 6 & 0 & 2 \\ 1 & 8 & 2 & 0 \end{bmatrix}$$

  The matrices D_{sort} and $D_{circles}$ corresponding to matrix D above are

$$D_{sort} = \begin{bmatrix} 0 & 1 & 3 & 4 \\ 0 & 3 & 6 & 8 \\ 0 & 2 & 4 & 6 \\ 0 & 1 & 2 & 8 \end{bmatrix}$$

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

- In the second phase, a matrix called D_{radius} is computed. All the entries along the main diagonal of this matrix are set to 1. Each element (i, j) with $i \neq j$ of this matrix provides a count of those nodes whose distance from node i is less than the distance of node j from node i. D_{radius} for example provided above is

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

These matrices find their use in generating the neighboring solution efficiently as explained in the next subsection.

5.3.5 Neighboring solution generation

We have employed two methods to generate neighboring solutions to ensure such solutions are the proper mix of diversity and quality. These two methods are based on the concept of critical distance introduced in [124]. It is defined as the largest distance among all distances between nodes and their nearest center. The node which is at the farthest distance from its nearest center is termed as the critical node. In the first method, to generate a neighboring solution Z' for a solution Z, first we copy the solution Z into Z'. Then we delete Q centers from Z'. These Q centers are deleted one-by-one in an iterative manner, where during each iteration, the center whose deletion has a minimum adverse effect on objective function is deleted from Z' (ties are broken arbitrarily). Deleting Q centers makes the Z' infeasible, and, hence, to restore the feasibility of the Z', Q centers need to be added. These Q centers are added one-by-one in the following manner: First, current critical distance is determined and then the node i and center j corresponding to this critical distance is identified. A node is randomly chosen to be a center from among non-center nodes which can reduce the current critical distance and added to Z', i.e., a new center is chosen using the following equation

$$new\_center = D_{circles}(i, rand(D_{radius}(i, j)))$$

This process is repeated till Q centers have been added. The value of Q is determined as follows

$$Q = \begin{cases} round(\frac{p}{2}) & \text{if } 5 \times p < n \\ round(\frac{p}{5}) & \text{otherwise} \end{cases} \tag{5.2}$$

The second method is a modified version of the method used by [124]. In this method, to generate a neighboring solution Z' for a solution Z, first we copy the solution Z into Z'. We choose one solution Z_1 (different from Z) from the population randomly. Then we add Q more centers to Z' one-by-one. Each center is added in the following manner: Like the first method, node i and center j corresponding to current critical distance is identified. Obviously, the solution quality can be improved only by reducing the current critical distance. To reduce the current critical distance, we find the set of candidate centers K which falls within the radius of the critical distance. These candidate centers K are at a distance less than the current critical distance from the critical node i.

The set K is determined from the following equation

$$K = \{D_{circles}(i, k), \text{ where } k = 1, \ldots, D_{radius}(i, j)\}$$

We compute the intersection of K and Z_1. If this intersection is nonempty, then we randomly select one node from set $K \cap Z_1$ and add it to Z'. If this intersection is empty, then we randomly select one node from set K and add it to Z'. This process is repeated until we have added Q centers. Davidović et al. [124] has always selected a node from K randomly and has not made use of another solution. The use of another solution is based on the fact that if a facility is located at a particular demand point in one good solution, then it is highly likely that a facility is located at the same demand point in many good solutions. Hence, the use of another solution helps in identifying potential nodes. Also, note that K is never empty as it always contains critical node i which is responsible for current critical distance.

Adding Q centers will reduce the critical distance, but makes the solution infeasible, hence we need to delete Q centers to restore its feasibility. The Q centers are deleted

one-by-one in the following manner. Among the $p + Q$ centers, we find the impact of each center on the solution quality and delete that center which has the least impact on the solution quality (ties are broken arbitrarily). This process is repeated until only p centers remain in the solution. The value of Q is determined using the same equation as in first method, i.e., Equation (5.2).

On the other hand, the value of Q in [124] is randomly chosen in the interval $[0, p]$, if $5 \times p$ is less than n and in the interval $[0, \frac{n-2.5p}{2.5}]$ otherwise, each time a new solution needs to be produced. So the value of Q varies in the method of [124]. If the solution Z, the original solution, and, the solution Z_1, the randomly chosen solution, are identical, then a collision occurs which is dealt in a way similar to the p-median problem as described in Section 2.3.4 of Chapter 2.

These two methods are used in a mutually exclusive way. The first method is utilized for generating a neighboring solution with probability p_{sel}, otherwise the second method is applied.

5.3.6 Other features

The rules for obtaining the scout bees are same as described in Section 2.3.5 of Chapter 2.

5.4 The IWO approach to the p-center problem

Several components of our IWO approach have been borrowed from our ABC approach (Section 5.3). Our IWO approach employs same solution encoding and same fitness function as our ABC approach. Subsequent subsections describe various features of our IWO algorithm for the p-center problem.

5.4.1 Initial solution

Initial weed solutions are produced randomly in the same manner as described in Section 5.3.3. The number of initial solutions n_i is a parameter of the algorithm.

5.4.2 Determination of number of seeds of a weed

Each weed solution, based on its own and the colony's lowest and highest fitness, produces a certain number of seed solutions. The number of such seed solutions lies in

the interval $[X_{min}, X_{max}]$, where X_{min} is the minimum possible number of seeds and X_{max} is the maximum possible number of seeds. However, unlike the traditional IWO algorithm where there is a linear mapping between a weed solution and the number of seed solutions it can produce, we have followed the strategy used in [72], where the total number of solutions present in the colony at a particular moment (N_c) are partitioned into $(X_{max} - X_{min}) + 1$ disjoint groups according to their fitness and the solutions belonging to the same group will produce the same number of seed solutions which lies in the interval $[X_{min}, X_{max}]$. The pseudo-code for determining the number of seeds is given in Algorithm 7.

Algorithm 7: Method for determining number of seeds

function `No_of_Seeds(`i`)`
Input: index i of a solution in the sorted solution list
Output: number of seeds that the solution corresponding to index i can produce
begin
  if $\left(i \leq \frac{N_c}{(X_{max}-X_{min})+1}\right)$ **then**
    return X_{max};
  else if $\left(i \leq \frac{2*N_c}{(X_{max}-X_{min})+1}\right)$ **then**
    return $X_{max} - 1$;
  ⋮
  ⋮
  else if $\left(i \leq \frac{(X_{max}-X_{min})*N_c}{(X_{max}-X_{min})+1}\right)$ **then**
    return $X_{min} + 1$;
  else
    return X_{min};

5.4.3 Production of seed of a weed

We have utilized the neighboring solution generation method of ABC algorithm (Section 5.3.5) for producing a seed solution corresponding to a weed solution.

5.4.4 Competitive exclusion

After some number of iterations, the number of solutions in the colony reaches the maximum allowable value n_{max} as new seed solutions are continuously produced and added to the colony. When such a situation arises, the competitive exclusion procedure begins discarding the solutions with poor fitness. The competitive exclusion procedure works as follows: Once all solutions in the colony have generated their designated number of seed solutions, all the solutions including newly generated seed solutions are sorted into non-increasing order as per their fitness and best n_{max} solutions are retained in the colony and remaining solutions are discarded. In this way, solutions having higher fitness only survive and reproduce.

The pseudo-code of our IWO approach is given in Algorithm 8, where n_c and n_i denote the number of weeds in the colony at any instance of time and the number of weeds in the colony at the beginning respectively. Sorting of weeds into non-increasing order based on their fitness is done by the function Sort_The_Weeds(). The number of seeds a weed W_i can produce as described in Section 5.4.2 is computed by the function No_of_Seeds(). Generate_Seed_Solution(W) is another function which returns a seed solution around the solution W (Section 5.4.3).

5.5 Computational results

Our ABC & IWO approaches have been implemented in C and executed on a Linux based Intel Core i5 2400 system with 4 GB memory running at 3.10 GHz. We have used the following parameter values for ABC algorithm in all our computational experiments : the number of employed bees (n_e) is 50, the number of onlooker bees (n_o) is 100, p_{onl} is set to 0.65, p_{sel} is set to 0.3, $limit$ is set to 50, Our ABC approach terminates after 100 iterations. The parameters of IWO are as follows: number of initial solutions (n_i) is 50, maximum number of solutions allowed in colony n_{max} is 200, p_{sel} is set to 0.35, X_{max} is set to 5, X_{min} is set to 1, maximum number of iterations is set to 50. All these parameter values have been chosen empirically based on a large number of trials. These parameter values yield good results for most instances. However, they are in no way optimal parameter values for all instances.

We have tested our approaches on same 40 OR-Library instances used in Chapter 2 as these instances have been used in the literature to test the performance of approaches

Algorithm 8: Pseudo-code of IWO approach

```
Generate n_i initial weed solutions W_1, W_2, ..., W_{n_i} randomly;
best := best solution among W_1, W_2, ..., W_{n_i};
n_c := n_i;
Sort_The_Weeds(n_c);
if n_c > n_max then
    n_c := n_max;
while Termination condition not satisfied do
    n_colony=n_c;
    for i := 1 to n_colony do
        seeds := No_of_Seeds(i);
        for j := 1 to seeds do
            n_c := n_c + 1;
            W_{n_c} := Generate_Seed_Solution(W_i);
            if W_{n_c} is better than best then
                best := W_{n_c};
    Sort_The_Weeds(n_c);
    if (n_c > n_max) then
        n_c := n_max;
return best;
```

for p-center problem also. The number of demand points in these instances vary from 100 to 900 and the number of centers from 5 to 200. We have compared the results obtained by ABC & IWO approaches on each instance with best values known so far, and, the best method known so far viz. BCOi [124]. Like [124], we have executed our approach once on each instance so as to allow a fair comparison with BCOi and other approaches. In Table 5.1, we have reported these results. The first column contains test problem number. The total number of nodes and the total number of centers are given in the next two columns. The best known values are reported under the column heading Best. These best known values are due to a single run of at least one of the methods among M-I, VNS, TS-1 and TS-2 [121], SS approach [123] and BCOi [124]. Next three columns present the values obtained by BCOi, ABC, and IWO during their solitary run. Columns 7-10 report the running time to reach the best value. Column

Table 5.1: The results of executing various approaches once on each of the 40 test problems of OR-Library

			Objective function value				Time(in seconds)			
Sno	N	P	Best	BCOi	ABC	IWO	Best(in lit)	BCOi	ABC	IWO
1	100	5	127	127	127	127	0.00	0.00	0.01	0.01
2	100	10	98	98	98	98	0.00	0.00	0.01	0.21
3	100	10	93	93	93	93	0.00	0.00	0.03	0.11
4	100	20	74	74	74	74	0.00	0.00	0.04	0.20
5	100	33	48	48	48	48	0.00	0.00	0.02	0.22
6	200	5	84	84	84	84	0.00	0.00	0.01	0.47
7	200	10	64	64	64	64	0.00	0.00	0.04	0.12
8	200	20	55	55	55	55	0.01	0.01	0.28	0.67
9	200	40	37	37	37	37	0.02	0.00	0.14	1.05
10	200	67	20	20	20	20	0.08	0.03	0.22	1.60
11	200	5	59	59	59	59	0.01	0.00	0.08	0.03
12	300	10	51	51	51	51	0.01	0.02	0.07	0.79
13	300	30	36	37	36	36	0.01	0.01	0.55	4.13
14	300	60	26	27	26	26	0.32	0.14	0.38	5.80
15	300	100	18	18	18	18	0.05	0.04	0.43	3.55
16	400	5	47	47	47	47	0.00	0.00	0.01	0.07
17	400	10	39	39	39	39	0.00	0.01	0.12	0.68
18	400	40	28	29	29	29	0.16	0.15	1.19	4.80
19	400	80	19	19	19	19	1.01	0.07	0.99	7.46
20	400	133	14	14	14	14	0.44	0.09	1.29	10.85
21	500	5	40	40	40	40	0.00	0.00	0.02	0.05
22	500	10	38	39	39	39	0.30	0.19	0.15	0.63
23	500	50	23	23	23	23	0.07	0.23	2.96	2.58
24	500	100	16	16	16	16	0.23	0.08	1.26	11.54
25	500	167	12	12	12	**11**	1.46	0.38	1.81	78.00
26	600	5	38	38	38	38	0.00	0.00	0.10	0.24
27	600	10	32	32	32	32	0.00	0.01	0.20	0.85
28	600	60	19	19	19	19	0.21	0.05	0.68	4.04
29	600	120	13	14	14	14	1.26	1.14	1.68	17.19
30	600	200	10	10	10	10	0.71	0.35	3.08	500.11
31	700	5	30	30	30	30	0.00	0.00	0.03	0.11
32	700	10	29	29	29	29	0.23	0.06	1.40	5.85
33	700	70	16	16	16	16	0.80	0.72	17.01	63.93
34	700	140	12	12	12	12	0.67	0.39	2.93	34.39
35	800	5	30	30	30	30	0.02	0.01	0.15	0.56
36	800	10	27	28	28	27	0.42	0.08	0.08	3.85
37	800	80	16	16	16	16	0.83	0.12	1.53	11.53
38	900	5	29	29	29	29	0.00	0.00	0.03	0.25
39	900	10	23	24	24	24	0.04	0.01	0.28	0.87
40	900	90	14	14	14	14	0.40	0.19	1.50	26.88

Table 5.2: Execution times (in seconds) of ABC, IWO and BCOi approaches

Sno	BCOi	ABC	IWO
1	1.00	0.14	0.34
2	2.00	0.51	1.24
3	2.00	0.52	1.35
4	2.00	0.64	1.51
5	1.00	1.29	3.18
6	2.00	0.29	3.18
7	2.00	1.02	2.56
8	2.00	3.09	7.96
9	2.00	3.87	10.00
10	2.00	9.04	23.28
11	2.00	0.40	1.07
12	2.00	1.59	3.99
13	2.00	9.31	23.52
14	3.00	11.78	30.29
15	6.00	28.48	73.00
16	1.00	0.54	1.36
17	2.00	2.15	5.46
18	70.00	20.61	52.82
19	4.00	26.12	66.56
20	4.00	62.33	160.02
21	2.00	0.74	1.84
22	2.00	2.74	6.91
23	4.00	38.75	97.78
24	5.00	49.64	123.31
25	4.00	122.25	296.14
26	2.00	1.10	2.05
27.	2.00	4.24	7.68
28	4.00	98.52	153.08
29	10.00	138.79	197.88
30	8.00	37.21	49.79
31	2.00	1.42	2.48
32	2.00	5.75	9.27
33	10.00	190.70	232.92
34	15.00	290.17	306.16
35	2.00	1.91	2.82
36	2.00	7.63	10.50
37	7.00	333.82	339.31
38	2.00	1.96	3.17
39	20.00	8.04	11.75
40	20.00	461.01	472.65

Table 5.3: Results of ABC & IWO approaches over 10 runs

		ABC					IWO				
Sno	Best_Known	Best	Avrg	Std_Dev	ABT	ATT	Best	Avrg	Std_Dev	ABT	ATT
1	127	127	127	0.00	0.005	0.138	127	127	0.00	0.012	0.339
2	98	98	98	0.00	0.083	0.510	98	98	0.00	0.096	1.242
3	93	93	93.7	0.46	0.142	0.524	93	93.5	0.49	0.110	1.351
4	74	74	74	0.00	0.072	0.635	74	74.2	0.60	0.168	1.505
5	48	48	48	0.00	0.029	1.298	48	48	0.00	0.196	3.179
6	84	84	84.2	0.40	0.097	0.285	84	84	0.00	0.385	0.713
7	64	64	64	0.00	0.088	1.017	64	64	0.00	0.193	2.558
8	55	55	55	0.00	0.320	3.097	55	55	0.00	0.728	7.961
9	37	37	37	0.00	0.169	3.869	37	37	0.00	0.989	10.000
10	20	20	20	0.00	0.235	9.043	20	20	0.00	1.685	23.279
11	59	59	59.2	0.40	0.069	0.400	59	59.1	0.30	0.119	1.079
12	51	51	51.2	0.40	0.280	1.585	51	51	0.00	0.636	3.994
13	36	36	36.7	0.46	1.671	9.312	36	36.1	0.30	3.881	23.520
14	26	26	26.2	0.40	2.970	11.784	26	26	0.00	4.908	30.285
15	18	18	18	0.00	0.414	28.480	18	18	0.00	3.811	72.996
16	47	47	47	0.00	0.011	0.538	47	47	0.00	0.073	1.360
17	39	39	39	0.00	0.299	2.151	39	39	0.00	0.521	5.458
18	28	29	29	0.00	2.323	20.613	29	29	0.00	4.450	52.824
19	19	19	19	0.00	1.687	26.122	19	19	0.00	7.006	66.560
20	14	14	14	0.00	1.048	62.329	14	14	0.00	10.591	160.022
21	40	40	40	0.00	0.046	0.735	40	40	0.00	0.146	1.843
22	38	39	39	0.00	0.147	2.742	39	39	0.00	0.476	6.911
23	23	23	23	0.00	2.478	38.749	23	23	0.00	6.838	97.781
24	16	16	16	0.00	1.558	49.637	**15**	15.9	0.30	15.154	123.315
25	12	**11**	11.5	0.50	24.822	122.248	**11**	11.4	0.49	38.269	296.135
26	38	38	38.1	0.30	0.232	1.096	38	38	0.00	0.208	2.054
27.	32	32	32	0.00	0.383	4.237	32	32	0.00	0.627	7.677
28	19	19	19	0.00	1.032	98.517	**18**	18.9	0.30	17.259	153.081
29	13	13	13.7	0.46	24.829	138.792	13	13.5	0.50	34.875	197.882
30	10	10	10	0.00	5.172	37.205	10	10	0.00	494.505	49.789
31	30	30	30	0.00	0.064	1.421	30	30	0.00	0.175	2.481
32	29	29	29.7	0.46	0.831	5.754	29	29.3	0.46	2.503	9.268
33	16	16	16.1	0.30	73.940	190.696	16	16	0.00	25.828	232.917
34	12	**11**	11.6	0.49	69.203	290.169	**11**	11.6	0.49	53.221	306.157
35	30	30	30.1	0.30	0.336	1.905	30	30.1	0.30	0.381	2.824
36	27	27	27.8	0.40	1.160	7.626	27	27.8	0.40	1.450	10.493
37	16	16	16	0.00	4.329	333.820	16	16	0.00	13.445	339.307
38	29	29	29	0.00	0.056	1.956	29	29	0.00	0.209	3.168
39	23	24	24	0.00	0.343	8.042	24	24	0.00	0.656	11.747
40	14	14	14	0.00	5.134	461.011	14	14	0.00	15.796	472.646

heading Best(in lit) report the running time from the literature to reach the best value. The last three columns report the time needed by BCOi, ABC, and IWO to reach the best objective function value. Data for M-I, VNS, TS-1 and TS-2, SS and BCOi approaches are taken from [124] where results were obtained on a Linux based Core 2 Duo system with 8GB RAM running at 2.66GHz. As this system is different from the system used to execute ABC & IWO approaches, therefore execution times can not be compared precisely. However, a rough comparison can always be made.

Table 5.1 clearly shows the effectiveness of our approaches in terms of solution quality. ABC algorithm obtained best known solution values in 35 out of 40 instances and IWO reached the best know solution values for 36 instances out of 40 in contrast to 33 for BCOi. Even, IWO was also able to improve the best known solution value for one instance (viz. pmed25), which is reported in bold face. From this table, it can also be seen that for those problem instances where our approaches fail to reach the best known values, they reached the second best value. However, our approaches are slower than other approaches in terms of time to reach the best solutions with IWO the slowest among the lot. In this table, we have compared the time to reach the best solution due to past precedences only. However, total execution time is a more standard and reliable measure of time taken by various approaches as best solution is returned only when the algorithm finishes execution and terminates no matter how fast the best solution is found. BCOi was executed for fixed amount of time on each instance that range from 1 second to 70 seconds depending on the instance. Table 5.2 reports the execution times of ABC & IWO approaches along with BCOi for each instance. Here the execution times of ABC & IWO approaches are average of 10 independent runs. From this table, we can see that our approaches, particularly ABC, are faster than BCOi on a number of instances.

To test the robustness of ABC & IWO approaches, we have executed each of them 10 independent times. The results are reported in Table 5.3. The first column contains the problem instance number, the second column represents the best known value. Next 3 columns report the best value, average value and standard deviation obtained by the ABC algorithm. Column 6 report the average time till best and column 7 report the average execution time of ABC algorithm. Similarly, we report the best value, average value, standard deviation, average time till best and average execution time for IWO algorithm in columns 8-12. From this table, it is clear that ABC algorithm is able to

improve the best known value for 2 instances and IWO algorithm obtained improved best known value for 4 instances, which are reported in bold face in this table. For ABC & IWO both, on around 66% of instances, average solution quality is same as the best solution quality which indicates ABC & IWO approaches are able to obtain best known values in every run. Even for remaining instances, average solution quality does not differ much from best solution quality.

5.6 Conclusions

In this chapter, we have proposed two metaheuristic approaches, viz. artificial bee colony algorithm and invasive weed optimization algorithm for the p-center problem. We have evaluated the performance of our proposed approaches against the best approach available in the literature, viz. BCOi [124] on the standard benchmark instances for this problem. Computational results show the effectiveness of our approaches in finding high quality solutions. However, our approaches are slower than BCOi on the majority of instances.

Chapter 6

Terminal assignment problem

6.1 Introduction

In the current and the next chapter, we will address three facility assignment problems. The first facility assignment problem considered is the terminal assignment problem which is addressed in this chapter. In the next chapter, we will deal with other two facility assignment problems.

Due to rapid growth of the internet, many new problems arose in the field of telecommunication network design and management. Terminal assignment (TA) problem is one such problem. The objective of the TA problem is to connect a given set of N terminals, each with a positive weight, to a given set of M concentrators, each with a fixed capacity, in such a way that the total cost of the network thus formed is minimum according to a given objective function. The assignment of the terminals to the concentrators is done under following two constraints: First, each terminal must be connected to one and only one concentrator, second, the sumtotal of weights of the terminals connected to a concentrator must not exceed the capacity of that concentrator [129] [130]. TA problem is solved in the literature with two different objectives: First, with the objective of minimizing the sumtotal of link costs alone and second with an objective which gives consideration to equitable distribution of loads among concentrators in addition to link costs. We have considered the latter objective. TA problem is proved $\mathcal{NP}$-Hard under the first objective [131]. The problem can be solved in polynomial time in the special case where all terminals have the same weight, and all concentrators have the same capacity. The TA problem is also $\mathcal{NP}$-Hard under the second objective as the

first objective can be considered as a special case of the second objective where no consideration is given to load on different concentrators. The problem is harder to solve under the second objective because we can not compute the individual cost of assigning a terminal to a concentrator a priori, i.e., before the complete solution is constructed. Many different approaches have been proposed in the literature to solve the TA problem. Abuali et al. [130] proposed a greedy heuristic and a greedy genetic algorithm for a restricted version of the problem where all concentrators have the same capacity. Khuri and Chiu [129] proposed another greedy heuristic and two penalty based genetic algorithms. Both Abuali et al. [130] and Khuri and Chiu [129] considered the first objective as mentioned above. Salcedo-sanz and Yao [132] considered for the first time the second objective where a hybrid Hopfield network based genetic algorithm is presented. Xu et al. [133] presented a tabu search based approach for TA problem. Bernardino et al. designed a local search genetic algorithm (LSGA) [134], a tabu search (TS) [135], a hybrid differential evolution algorithm (HDE) [136], an improved hybrid differential evolution algorithm with a multiple strategy (MHDE) [137] and a discrete differential evolution algorithm (DDE) [138] for solving the TA problem with second objective. The DDE algorithm is based on discrete differential evolution model proposed by Pan et al. [52]. DDE algorithm provides the better results in comparison to LSGA, TS, and MHDE [138]. We have proposed an artificial bee colony algorithm based approach for solving the TA problem. We have compared our ABC approach with 4 best approaches from the literature, viz. DDE, MHDE, TS and LSGA. In comparison to these approaches, our approach not only obtains the solution of better quality but is also faster.

The remaining part of this chapter is organized as follows: In Section 6.2, we describe the TA problem formally. Section 6.3 describes our ABC approach for the TA problem. Section 6.4 reports the computational results and compares our approach with state-of-the-art approaches available in the literature. Finally, Section 6.5 contains some concluding remarks.

6.2 Formal description

Given a set of N terminals $T = \{T_1, T_2, \ldots, T_N\}$, a set of M concentrators $C = \{C_1, C_2, \ldots, C_M\}$, a weight or capacity requirement W_i associated with each termi-

nal $T_i \in T$, and, a capacity X_j associated with each concentrator $C_i \in C$. The weights of terminals are such that $W_i < \min(X_1, X_2, \ldots, X_M)\ \forall\ T_i \in T$. The TA problem seeks an assignment of terminals to concentrators without violating the capacity constraint of concentrators such that the considered objective function is optimized. Hence, any feasible solution must satisfy the following two constraints:

$$\sum_{j=1}^{M} z_{ij} = 1 \quad \forall T_i \in T \tag{6.1}$$

$$\sum_{i=1}^{N} W_i z_{ij} \leq X_j \quad \forall C_j \in C \tag{6.2}$$

Where binary variables z_{ij} indicate whether terminal T_i is assigned to concentrator C_j ($z_{ij} = 1$) or not ($z_{ij} = 0$). The Equation (6.1) states that each terminal can be assigned to one and only one concentrator, whereas Equation (6.2) states that capacity constraint of none of the concentrators should be violated.

We have considered the same objective function for TA problem as used in [138]. This objective function considers the two factors:

- The total number of terminals assigned to each concentrator
- The distance between the terminals and their assigned concentrators

The objective is to minimize the Equation (6.5). The Equation (6.3) and Equation (6.4) define terms needed for defining the objective function.

$$Total_{C_j} = \sum_{i=1}^{N} z_{ij} \quad \forall C_j \in C \tag{6.3}$$

$$Bal_{C_j} = \begin{cases} 10 & if\ (Total_{C_j} = round(\frac{N}{M}) + 1) \\ 20\times \mid (round(\frac{N}{M}) + 1 - Total_{C_j}) \mid & otherwise \end{cases} \quad \forall C_j \in C \tag{6.4}$$

$$objective\ function\ value = 0.9 \times \sum_{j=1}^{M} Bal_{C_j} + 0.1 \times \sum_{i=1}^{N}\sum_{j=1}^{M} D_{ij} z_{ij} \tag{6.5}$$

Where D_{ij} is the distance between terminal T_i and concentrator C_j.

6.3 ABC approach for the TA problem

This section presents our ABC approach for TA problem. Salient features of our proposed approach are described in following subsections.

6.3.1 Solution encoding

To encode a solution, we have used the terminal based representation proposed in the literature [138]. The value represented by position i specifies the concentrator to which the terminal i is assigned. Figure 6.1 explains this representation with the help of an example where there are 10 terminals and 3 concentrators. In this figure, terminals 1, 3 and 6 are assigned to concentrator 1, terminals 2, 4, 7 and 9 are assigned to concentrator 2, and, terminals 5, 8 and 10 are assigned to concentrator 3.

1	2	1	2	3	1	2	3	2	3

Figure 6.1: Solution representation

6.3.2 Initial solution generation

Each initial solution is obtained by using a method which is partially greedy and partially random. In this method, with probability p_{asn}, the terminals are greedily assigned to the nearest available concentrator. Here, the availability refers to the remaining capacity of the concentrator to serve the terminal capacity requirement. If the capacity requirement is not satisfied then the terminal is assigned to the next nearest concentrator in case it satisfies the capacity requirement; otherwise, this process is repeated until an available concentrator is found or none exists. With probability $1 - p_{asn}$, terminals will be assigned to the available concentrators randomly. The algorithm iterates through this process until all the terminals are assigned. The terminal to be assigned next is selected randomly.

In case no available concentrator exists for a terminal then the solution is infeasible. This solution is discarded, and we start afresh in a bid to generate a feasible solution. If we are not able to generate a feasible solution even after three attempts, then the last infeasible solution is included in the population, but its fitness is penalised using a penalty term as explained in Section 6.3.5. In this infeasible solution, terminals which

can not be assigned to any available concentrator are assigned to some randomly chosen concentrator.

6.3.3 Generation of neighboring solution

To generate a solution $S_i^{'}$ in the neighborhood or vicinity of a solution S_i, each terminal in S_i is reassigned with probability p_{rand} using a greedy approach. In the greedy approach, the terminals are assigned to the nearest available concentrators. S_i is replaced with the neighboring solution $S_i^{'}$ if the fitness of $S_i^{'}$ is better than S_i. In case the neighboring solution is infeasible then we discard the solution. This can happen only when the original solution is infeasible.

The pseudo-code for generating a neighboring solution $S_i^{'}$ in the vicinity of a solution S_i is given in Algorithm 9, where $U_{01}()$ is a function that returns a random number between 0 and 1.

Algorithm 9: generate_neighboring_solution(S_i)

```
for each terminal T_j ∈ T do
    p ← U_01();
    if p ≤ p_rand then
        Assign T_j to the nearest available concentrator in S_i';
    else
        Assign T_j to the same concentrator in S_i' as in S_i;
return S_i';
```

6.3.4 Selecting a food source for an onlooker bee

The binary tournament selection method is used for selecting a food source for an onlooker bee, where the probability of selecting the candidate with better fitness is p_{onl}.

6.3.5 Fitness of a solution

To evaluate the fitness of a solution, we have used the same fitness function as used in [138]. This fitness function is a modification of the objective function given in

the Section 6.2. The fitness function adds a penalty term called *penalization* to the objective function for infeasible solutions. The *penalization* is computed as follows:

$$penalisation = \begin{cases} 0 & \text{if solution is feasible} \\ 500 & \text{otherwise} \end{cases} \tag{6.6}$$

$$fitness = objective\ function\ value\ +\ penalisation. \tag{6.7}$$

This fitness needs to be minimized.

6.3.6 Other features

If a solution associated with an employee bee does not improve for *limit* number of iterations then this employee bee becomes a scout.

6.3.7 Local search

In order to improve the quality of the solution generated by our algorithm, we have applied two local search methods one after the other on the best solution obtained through our ABC algorithm. In the first local search method, terminals are considered one-by-one to check if, it is assigned to the concentrator having the maximum load, and if so, it is considered for shifting to a less loaded concentrator in case this shifting satisfy the capacity constraints and improves the objective function. For this shifting, concentrators are considered in non-decreasing order of their load. The second local search method is based on swapping the concentrators allocated to two terminals. In this method, two terminals allocated to different concentrators are chosen randomly and their respective concentrators are swapped if doing so improves the quality of the solution. This job of swapping continues as long as there is an improvement in solution quality.

6.4 Computational results

Table 6.1: Solution quality of various approaches

	LSGA			TS			MHDE			DDE			ABC			ABC-LS		
prob	Best	Avg	SD	Best	Avg	SD	Best	Avg	SD	Best	Avg	SD	Best	Avg	SD	Best	Avg	SD
1	65.63	65.63	0.00	65.63	65.63	0.00	65.63	65.63	0.00	65.63	65.63	0.00	**65.63**	**65.63**	0.00	**65.63**	**65.63**	0.00
2	134.65	134.65	0.00	134.65	134.65	0.00	134.65	134.65	0.00	134.65	134.65	0.00	**133.41**	**133.41**	0.00	**133.41**	**133.41**	0.00
3	270.26	270.69	0.23	270.26	270.76	0.30	270.26	270.75	0.15	270.26	270.47	0.22	279.94	281.17	0.66	277.27	281.10	1.21
4	286.89	286.99	0.13	286.89	287.93	0.75	286.89	287.17	0.14	286.89	286.89	0.00	**286.61**	**286.65**	0.09	**286.61**	**286.65**	0.09
5	335.09	335.99	0.60	335.09	335.99	0.59	335.09	336.55	0.39	335.09	335.26	0.17	**335.07**	336.24	0.32	**335.07**	336.24	0.32
6	371.12	371.68	0.24	371.12	372.44	0.45	371.12	373.19	0.42	371.12	371.38	0.22	374.55	378.44	1.38	373.44	378.06	1.55
7	401.21	402.41	0.50	401.29	403.25	0.73	401.21	403.61	0.33	401.21	401.62	0.28	**399.85**	**400.59**	0.39	**399.78**	**400.41**	**0.31**
8	563.19	564.94	0.52	563.34	564.5	0.54	563.19	572.04	0.76	563.19	564.07	0.38	596.65	601.85	2.53	590.06	597.86	3.32
9	642.83	646.52	0.84	642.86	644.18	0.48	642.83	648.46	0.48	642.83	643.96	0.46	687.52	690.71	1.65	668.51	676.16	3.02

Table 6.2: Time taken in seconds to reach the best solution by various algorithm

prob	LSGA	TS	MHDE	DDE	ABC	ABC-LS
1	<1s	<1s	<1s	<1s	<1s	<1s
2	<1s	<1s	<1s	<1s	<1s	<1s
3	<1s	<1s	<1s	<1s	<1s	<1s
4	<1s	<1s	<1s	<1s	<1s	<1s
5	<1s	<1s	<1s	<1s	<1s	<1s
6	1s	<1s	<1s	<1s	<1s	<1s
7	1s	1s	2s	<1s	<1s	<1s
8	7s	1s	10s	2s	<1s	1s
9	7s	2s	15s	3s	<1s	2s

We implemented our approach in C and executed on an Intel Core 2 Duo (E8400) system with 2 GB RAM running at 3.0 GHz under Fedora 12 release. In all our computational experiments, the number of employee bees (n_e) is taken to be 50 and the number of onlooker bees (n_o) is taken to be 100. We have used p_{rand}=0.2, p_{asn} =0.85, p_{onl}=0.85, $limit$=500 in all our experiments. Our ABC approach terminates after 1500 iterations. All these parameter values are chosen empirically, after executing the algorithm multiple times. In order to test the performance of our approach, we have used the 9 benchmark instances available in the literature [138]. On each instance, we have executed our approach 40 independent times. We compare the results of our approach with those of LSGA [134], TS [135], MHDE [137] and DDE [138]. Results of these four approaches are taken from [138]. We have reported the results of our ABC approach with and without the use of local search.

Table 6.1 compares the different approaches in terms of best and average solution quality and standard deviation of solution values on each of the 9 benchmark instances. In this table, results of our approach without local search is reported under the column heading ABC and with local search is reported under the column heading ABC-LS. Results are shown in bold whenever they are as good as or better than previous 4 approaches. From this table, it can be clearly seen that performance of our approaches is more-or-less comparable to these 4 state-of-the-art approaches. Our approaches obtain new best solution values for 4 instances. Further, local search has only small impact on solution quality as the solution obtained through ABC algorithm is improved

Table 6.3: Influence of parameter settings on solution quality

Parameter	Value	Problem 4		Problem 5	
		Best	Avg	Best	Avg
n_e	25	286.61	286.87	335.07	336.19
	50	**286.61**	**286.65**	**335.07**	**336.23**
	75	286.61	286.65	335.07	336.13
	100	286.61	286.62	335.07	336.12
n_o	50	286.61	286.67	335.69	336.44
	75	286.61	286.68	335.69	336.41
	100	**286.61**	**286.65**	**335.07**	**336.23**
	125	286.61	286.69	335.56	336.20
p_{asn}	0.75	286.61	286.79	335.71	336.29
	0.8	286.61	286.76	335.07	336.19
	0.85	**286.61**	**286.65**	**335.07**	**336.23**
	0.9	286.61	286.67	335.07	336.31
	0.95	286.61	286.68	335.07	336.27
p_{onl}	0.75	286.61	286.68	335.07	336.25
	0.8	286.60	286.67	335.07	336.21
	0.85	**286.61**	**286.65**	**335.07**	**336.23**
	0.9	286.61	286.70	335.69	336.30
	0.95	286.61	286.66	335.07	336.11
p_{rand}	0.1	286.61	286.71	335.84	336.44
	0.15	286.61	286.74	335.69	336.33
	0.2	**286.61**	**286.65**	**335.07**	**336.23**
	0.25	286.61	286.69	335.07	336.16
	0.3	286.61	286.77	335.69	336.32
limit	300	286.61	286.68	335.52	336.12
	400	286.61	286.62	335.19	336.16
	500	**286.61**	**286.65**	**335.07**	**336.23**
	600	286.61	286.72	335.07	336.20

only slightly by the local search, and that too, mostly in case of large instances only.

Table 6.2 reports the time taken by various approaches to reach the best solution. Actually, [138] reported the time taken by various approaches to reach the best solution and not the total execution times. Hence, we are also reporting the time taken by various approaches to reach the best solution only in the same format as in [138]. Data for LSGA, TS, MHDE, DDE approaches are taken from [138]. As the four previous approaches were executed on an Intel Core Duo (T2300) based system which is different from the system used to execute our ABC and ABC-LS approaches, therefore times can not be compared precisely. However, a rough comparison can always be made. Even after compensating for the difference in processing speed, we can safely say that our approaches are faster on large instances. Our approach without local search requires less than 1 second to reach the best solution on all 9 instances . For instances 8 and 9, our approach with local search is taking 1 second and 2 seconds respectively to reach the best solution, whereas on the remaining instances it also takes less than 1 second only.

To investigate the influence of parameter settings on solution quality, we have taken two different instances, viz. Problem 4 and 5. We have varied all the parameters one by one while keeping all other parameters unchanged. In doing so, all other parameters were set to their values reported at the start of this section. The results are reported in Table 6.3. Values in bold in this table show the results with original parameter values which are used in all the experiments involving our approach. From this table, it can be seen that values chosen by us provide either the best results or results which are very close to best results. In those cases where we have not got the best results with chosen parameter values, the parameter values chosen have provided best results on some other instances not included in this table.

6.5 Conclusions

In this chapter, we have proposed an ABC algorithm based approach for the TA problem and compared it with the best methods proposed in the literature. The results show the performance of the ABC algorithm is comparable with these methods. ABC algorithm provides good quality solutions in lesser execution times for the majority of

the instances. A local search is also tried to improve the best solution obtained through ABC algorithm but it has a little impact only.

Chapter 7

Ring loading problems

7.1 Introduction

Synchronous Optical NETworking (SONET) in USA and Canada, Synchronous Digital Hierarchy (SDH) in the rest of the world are the current standards for transmitting and multiplexing high speed signals over optical fibre communication networks. Basically, SONET/SDH enforce a ring based topology where nodes are connected via a ring of optical fibre cables. Each node is equipped with an Add-Drop-Multiplexer (ADM) that acts as an interface between the node and the ring and that performs the following functions

- It sends the message originating at its associated node over the ring
- It receives the message meant for its associated node over the ring and removes that message from the ring
- It relays all other messages

SONET/SDH rings are bi-directional, i.e., a message can be transmitted in either direction (clockwise or counter-clockwise) over the ring, and, the messages routed through clockwise direction compete with those messages which are routed through counter-clockwise direction for the common bandwidth. In addition to the type of optical fibre cable, the bandwidth available along any edge of SONET/SDH ring depends on the ADM [139]. The amount of data transmitting through an edge in either direction at a particular instant is called its load at that instant. Obviously, the load on any edge can not exceed the available bandwidth. Weighted Ring Edge-Loading Problem (WRELP)

is an important problem in this context that seeks to minimize the maximum load on any edge. Given a set of communication demands between various pairs of nodes, the WRELP consists in routing the demands on the ring in either clockwise or counter-clockwise direction so that the maximum load over all the edges is minimized.

IEEE 802.17 Standard for Resilient Packet Ring (RPR) combines the benefits of SONET/SDH and Ethernet networks for significantly enhancing the performance of optical fibre ring networks with regard to data traffic [140, 141, 142]. Like SONET/SDH, RPR rings are also bidirectional. However, unlike SONET/SDH where there is a single bi-directional ring, RPR consists of two distinct uni-directional rings (one clockwise and another counter-clockwise) each with its own bandwidth, and, the messages sent through clockwise ring do not compete with those messages which are sent through the counter-clockwise ring for the common bandwidth. So bi-directionality in RPR is achieved by making use of these two rings while sending the messages. Clearly, in RPR, we have to deal with directed edges or arcs, where an arc, depending on its direction, belongs to the clockwise or counter-clockwise ring. The amount of data transmitting through an arc at a particular instant is called its load at that instant. Weighted Ring Arc-Loading Problem (WRALP) in RPR is equivalent of WRELP in SONET/SDH. Given a set of communication demands between various pairs of nodes, the WRALP problem consists in routing the demands either through the clockwise ring or counter-clockwise ring so that the maximum load over all the arcs of both the rings is minimized.

Depending on whether demands can be split or not, there are two variants of WRELP and WRALP. In the first variant, demands can be split into two parts with each part routed through a different direction, whereas as the second variant does not allow splitting of demands, i.e., each demand has to be routed in-toto through one of the two directions. The first variant can be solved in polynomial time, whereas the latter variant is $\mathcal{NP}$-Hard. For those interested in the first variant may refer to [142, 143, 144, 145, 146]. In this chapter, we have considered the second variant only. WRELP and WRALP with non-split demands are referred to as non-split WRELP and non-split WRALP respectively in the literature. Hereafter, in this chapter, WRELP and WRALP will refer to their respective non-split variants only even if we don't use the qualifier "non-split".

Cosares and Saniee [147] introduced non-split WRELP and proved its $\mathcal{NP}$-hardness. Schrijver et al. [148] proposed a highly efficient greedy heuristic which returns a solution that can exceed the optimal by at most $\frac{3}{2}$ times the maximum demand and which performs much better in practice. Extending this work, Khanna [149] developed a polynomial time approximation scheme for this problem. Dell et al. [150] presented efficient lower and upper bounding procedures, and a branch-and-bound based exact algorithm. The non-split WRALP was introduced and proved $\mathcal{NP}$-hard by Yuan et al.[142] and further studied in [146, 151].

Among the metaheuristics techniques for WRELP and WRALP, Karunanithi et al. [152] proposed a genetic algorithm for WRELP and solved the small instances of WRELP. Kim et al. [153] presented different variations of Ant Colony Optimization (ACO) algorithm for WRELP. Bernardino et al. [154, 155, 156, 157, 158, 159] proposed several evolutionary algorithms based approaches and a tabu search based approach for WRELP and several evolutionary algorithms and swarm intelligence based approaches for WRALP. In particular, Bernardino et al. [158] describes an artificial bee colony algorithm based approach for WRALP. Bernardino et al. [160] presented a genetic algorithm (GA), a hybrid differential evolution algorithm (HDEM) and a hybrid discrete particle swarm optimization algorithm (HDPSO) for WRELP and WRALP and compared the performance of these three approaches on both the problems with best performing previously proposed approaches mentioned above. The HDEM performed the best followed by ABC approach of [158] and HDPSO on both the problems. It is to be noted that ABC approach was presented in [158] for WRALP only. However, Bernardino et al. [160] presented the results of ABC approach for WRELP also.

In this chapter, we present artificial bee colony (ABC) algorithm based hybrid approaches for WRELP and WRALP. The best solution obtained through ABC algorithm is improved further by two local searches which are applied one after the other. Except for the values of some parameters, the ABC algorithm is same for WRELP and WRALP, but the two local searches differ according to the problem. As described in Section 7.3, our ABC algorithm is altogether different from the one proposed in [158]. We have compared our hybrid approaches with GA, HDEM, HDPSO approaches proposed in [160] and ABC approach proposed in [158] on the benchmark instances available in the literature. Computational results on the standard benchmark instances show the superiority of our proposed approaches over these approaches.

The remaining part of this chapter is organized as follows: Section 7.2 defines the two ring loading problems in a formal manner. Our hybrid ABC approaches for WRELP and WRALP are presented in Section 7.3. In Section 7.4 the computational results and a comparative analysis of our hybrid approaches vis-à-vis state-of-the-art approaches available in the literature are provided. Finally, Section 7.5 outlines some concluding remarks.

7.2 Problem formulation

Given a n node bi-directional ring $R_n = \{n_1, n_2, \ldots, n_n\}$ such that $e_i =< n_i, n_{i+1} > \forall i \in \{1, 2, \ldots, n-1\}$ and $e_n =< n_n, n_1 >$ are edges of this ring. Each edge $e_i =< n_i, n_{i+1} >$ corresponds to two directed edges or arcs represented by ordered pairs $e_i^+ = (n_i, n_{i+1})$ and $e_i^- = (n_{i+1}, n_i)$. Likewise $e_n =< n_n, n_1 >$ corresponds to two arcs $e_n^+ = (n_n, n_1)$ and $e_n^- = (n_1, n_n)$. The direction $n_1, n_2, \ldots, n_n$ corresponds to clockwise direction and $n_n, n_{n-1}, \ldots, n_1$ corresponds to counter-clockwise direction. So e_i^+, $\forall i \in \{1, 2, \ldots, n\}$ are arcs in clockwise direction, whereas e_i^-, $\forall i \in \{1, 2, \ldots, n\}$ are arcs in counter-clockwise direction. A set of m demands is given where each demand j is represented by a triple (s_j, d_j, w_j) where s_j is the source node, $d_j \neq s_j$ is the destination node and $w_j > 0$ is the weight associated with the demand j, which can be interpreted as the size of the data to be transmitted or amount of traffic generated to fulfil this demand. Demands can not be split, i.e., each demand need to be routed in-toto in one of the two directions. By associating binary variables x_j with each demand j such that $x_j = 1$ means demand j is routed through clockwise direction and $x_j = 0$ means demand j is routed through counter-clockwise direction, the load $L(e_i)$ on an edge e_i can be determined as follows: Clearly, $L(e_i) = L(e_i^+) + L(e_i^-)$ where

$L(e_i^+) = \sum_{j=1}^{m} f(i,j) \times w_j$

and

$L(e_i^-) = \sum_{j=1}^{m} g(i,j) \times w_j$

The value of $f(i,j)$ and $g(i,j)$ can be determined in the following manner:

$$f(i,j) = \begin{cases} 0 & \text{if } x_j = 0 \\ 1 & \text{if } ((i < n) \text{ and } (x_j = 1) \text{ and } ((s_j \leq n_i \text{ and } d_j > n_i) \text{ or } (s_j > d_j \text{ and } s_j > n_{i+1} \text{ and } d_j > n_i))) \\ 1 & \text{if } ((i = n) \text{ and } (x_j = 1) \text{ and } (s_j > d_j \text{ and } s_j > n_1 \text{ and } d_j \geq n_1)) \\ 0 & \text{otherwise} \end{cases}$$

and

$$g(i,j) = \begin{cases} 0 & \text{if } x_j = 1 \\ 1 & \text{if } ((i < n) \text{ and } (x_j = 0) \text{ and } ((s_j > n_i \text{ and } d_j \leq n_i) \text{ or } (s_j < d_j \text{ and } s_j < n_i \text{ and } d_j \leq n_i))) \\ 1 & \text{if } ((i = n) \text{ and } (x_j = 0) \text{ and } (s_j < d_j \text{ and } s_j < n_i \text{ and } d_j \leq n_i)) \\ 0 & \text{otherwise} \end{cases}$$

The WRELP seeks a route for each of these m demands such that the maximum load over all the edges is minimized. In other words, the objective of WRELP is to find the values of m binary variables $x_1, x_2, \ldots x_m$ in such a manner that minimizes $max(L(e_1), L(e_2), \ldots, L(e_n))$.

The WRALP seeks a route for each of these m demands such that the maximum load over all the arcs is minimized. In other words, the objective of WRELP is to find the values of m binary variables $x_1, x_2, \ldots x_m$ in such a manner that minimizes $max(\ell^+, \ell^-)$, where

$\ell^+ = max(L(e_1^+), L(e_2^+), \ldots, L(e_n^+))$

and

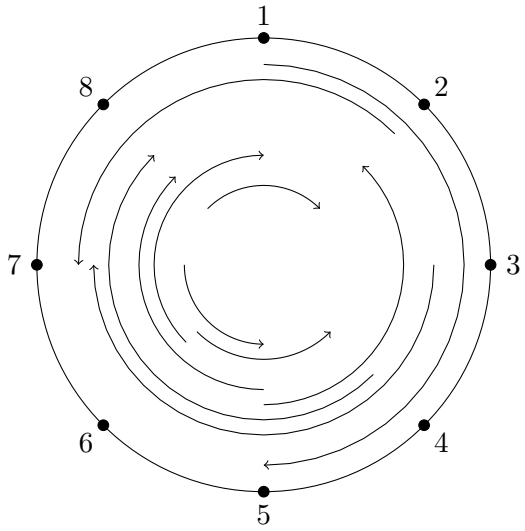
$\ell^- = max(L(e_1^-), L(e_2^-), \ldots, L(e_n^-))$.

For both WRELP and WRALP, the size of the search space is 2^m as each of m demands can be routed through one of the two possible directions. Please note that the size of the search space is independent of the number of nodes in the ring and depends only on the number of demands.

To illustrate WRELP and WRALP, let us consider the 8 nodes ring shown in Figure 7.1. Suppose 10 communication demands occur between different pairs of nodes in this ring. These demands, along with their route and values of their associated variables x_i in a routing scheme (a feasible solution) are shown below:

$(1, 5, 4)$	*clockwise*	$x_1 = 1$
$(2, 7, 3)$	*counter-clockwise*	$x_2 = 0$
$(3, 7, 2)$	*clockwise*	$x_3 = 1$
$(4, 8, 5)$	*clockwise*	$x_4 = 1$
$(5, 2, 7)$	*counter-clockwise*	$x_5 = 0$
$(5, 8, 3)$	*clockwise*	$x_6 = 1$
$(6, 1, 5)$	*clockwise*	$x_7 = 1$
$(6, 4, 4)$	*counter-clockwise*	$x_8 = 0$
$(7, 5, 8)$	*counter-clockwise*	$x_9 = 0$
$(8, 2, 3)$	*clockwise*	$x_{10} = 1$

Figure 7.1: Illustrating WRELP and WRALP



The route for each demand is also shown in Figure 7.1. In this routing scheme, loads on various arcs and edges are as follows:

$$\begin{array}{lll} L(e_1^+) = 7 & L(e_1^-) = 3 & L(e_1) = 10 \\ L(e_2^+) = 4 & L(e_2^-) = 7 & L(e_2) = 11 \\ L(e_3^+) = 6 & L(e_3^-) = 7 & L(e_3) = 13 \\ L(e_4^+) = 11 & L(e_4^-) = 11 & L(e_4) = 22 \\ L(e_5^+) = 10 & L(e_5^-) = 12 & L(e_5) = 22 \\ L(e_6^+) = 15 & L(e_6^-) = 8 & L(e_6) = 23 \\ L(e_7^+) = 13 & L(e_7^-) = 3 & L(e_7) = 16 \\ L(e_8^+) = 8 & L(e_8^-) = 3 & L(e_8) = 11 \end{array}$$

If the above routing scheme is used for WRELP then objective function value is 23. If the above routing scheme is used for WRALP then objective function value is 15 ($max(\ell^+, \ell^-)$, where $\ell^+ = 15$, $\ell^- = 12$).

7.3 Hybrid ABC approaches for WRELP and WRALP

We have developed hybrid approaches combining artificial bee colony (ABC) algorithm with local search procedures for WRELP and WRALP. Except for fitness function and values of some parameters, the ABC algorithm is same for both the problems. The local search procedure consists of two local searches that are applied one after the other.

These two local searches vary according to the problem. The local search procedure is applied only to the best solution obtained through ABC algorithm in a bid to further improve its quality. Hereafter, our hybrid approach for WRELP will be referred to as HABC-E, whereas our hybrid approach for WRALP will be referred to as HABC-A.

Following subsections describe other salient features of proposed approaches.

7.3.1 Solution encoding

We have used a bit vector of length m to represent a solution, where m is the total number of communication demands. A value of 1 at the i^{th} position indicates that demand i to be routed in the clockwise direction, whereas a value of 0 at the same place indicates that demand i to be routed in the counter-clockwise direction. Bernardino et al. [158, 160] also used the same encoding.

7.3.2 Fitness

We have used the objective function as the fitness function. So, HABC-E uses the objective function of WRELP as the fitness function, whereas HABC-A uses the objective function of WRALP as the fitness function. As WRELP and WRALP are minimization problems, so for these problems, a lower value of the fitness function indicates a more fit solution.

7.3.3 Initial employed bee solutions

Among the initial employed bee solutions, the first solution is generated by following the shortest path strategy where each demand is routed through the direction where it has to traverse the lesser number of links in comparison to the other direction (ties are broken arbitrarily). All other initial employed bee solutions are generated by following either a purely random strategy or a strategy that is a mix of greediness and randomness. The first strategy is used with probability ρ_{ir}, otherwise the second strategy is used. In the first strategy, each demand is routed uniformly at random in one of the two directions. In the second strategy, each demand is routed through the shortest path with probability ρ_{ig}, otherwise it is routed uniformly at random in one of the two

directions. ρ_{ir} and ρ_{ig} are two parameters whose values need to be determined empirically. The values for these parameters are chosen in such a manner so that the initial population of employed bee solutions is a proper mix of greediness and randomness.

The ABC approach of [158] generates initial employed bee solutions either in a completely random manner or in a completely deterministic manner. The deterministic strategy was again based on shortest path algorithm.

7.3.4 Policy used by onlookers to select a food source

We have used the binary tournament selection method to select a food source for each onlooker bee where the candidate with better fitness is selected with probability ρ_o.

The ABC algorithm of [158] assigns $P_i \times NO$ onlookers to a food source i, where $P_i = \frac{\sum_{j=1}^{NE} F_j - F_i}{\sum_{j=1}^{NE} F_j}$, F_i is the fitness of food source i, and NE is the number of employed bees and NO is the upper limit on the number of onlookers that can be sent to any food source. Usually, $\sum_{j=1}^{NE} F_j$ is much larger than the fitness of any solution, and hence, P_i is near to 1 for all $i = 1, \ldots, NE$. NO is taken to be greater than NE in [158], so a large number of onlookers are deputed for every solution in their approach. In contrast, we have used the total number of onlookers for all food sources to be twice as much as there are food sources.

7.3.5 Neighboring solution generation

Our neighboring solution generation method is based on the fact that if a demand is routed through a particular direction in one good solution; then it is highly likely that this demand is routed through the same direction in many good solutions. Hence, to generate a new solution X' in the neighborhood of a solution X, another solution Y is utilized. The solution Y is chosen randomly from all employed bee solutions other than X. Then to create X', each demand is considered one-by-one and it is routed in X' through the same direction as in Y with small probability ρ_{ns}, otherwise it is routed through the same direction as in X. Pseudo-code of neighboring solution generation process is given in Algorithm 10, where $X[i]$ refers to the i^{th} element (bit) of solution X which is a bit vector.

We have applied the afore-mentioned neighboring solution generation method in both employed and onlooker bee phases. Bernardino et al. [158] used different neighboring solution generation methods in employed and onlooker bee phases. Unlike our

Algorithm 10: Pseudo-code of neighboring solution generation process

```
Input: A solution X
Output: A new solution X' in the neighborhood of X
Select a solution Y randomly from all employed bee solutions other than X;
for i ← 1 to m do
    if (u_01<ρ_ns) then
        X'[i] ← Y[i];
    else
        X'[i] ← X[i];
return X';
```

neighboring solution generation method which does not use any form of local search, these methods do a partial local search around a solution to generate its neighboring solution. Hence, all these methods are computationally much more expensive than our method. In the employed bee phase, two neighboring solution generation methods are used. In the first method called "Exchange Direction", some demand pairs are randomly selected, and each such pair is checked one-by-one to see whether the exchange of directions between the two demands in the pair can improve the original solution. If more than one such exchanges are possible, then the exchange which results in a solution of least cost is performed and the solution thus obtained replaces the original solution. If no such exchange exists then, the original solution does not change. The second method called "Exchange Max Arc" randomly selects some demands routed through the arc having the highest load and tries to flip the direction of each of these demands one-by-one so as to improve the solution. Again the best improvement strategy is followed like the previous method. In onlooker bee phase, the neighboring solution generation method tries to improve a solution by changing the direction of a randomly chosen demand in the solution. With probability 0.5, the direction of the chosen demand is set to the shortest path; otherwise, the direction of the chosen demand is set to its direction in the best solution. If the solution improves, the methods stop; otherwise, it repeats itself. The method stops only after some fixed number of unsuccessful attempts.

7.3.6 Other features

If an employed bee solution X has not improved over S_{it} number of consecutive iterations, then it is assumed to be locally optimal and replaced with a new solution X'. However, instead of generating X' like the initial population members, which is usually the case with most ABC algorithms including the one described in [158], we have generated X' by perturbing X. For perturbing X to generate X', each demand is considered one-by-one and its direction in X is flipped with probability ρ_s. We have followed this perturbation strategy because solution thus generated retains major part of the original highly fit solution, and as a result, it is expected to be of better quality in comparison to a solution generated in a manner similar to initial population members.

Unlike the traditional ABC algorithm where there is an upper limit of only one scout in an iteration, we have not imposed any limit on the number of employed bee solutions replaced in an iteration, i.e., the number of scouts. This number can be more than one or can be zero depending on how many employed bee solutions in an iteration has not improved over last S_{it} number of consecutive iterations. This is also different from ABC approach of [158] where, in every iteration, the number of scouts is equal to 10% of the number of employed bees. The solutions for these scouts are created in the same manner as the initial population members. Instead of replacing those employed bee solutions which have not improved since long, these scout bee solutions replace worst employed bee solutions in case scout bee solutions are better. However, this is a severe flaw because if an employed bee solution is locally optimal, it will forever remain in the population and waste computational efforts without offering any opportunity for further improvement.

7.3.7 Local search procedure

As mentioned already, our local search procedure is composed of two local searches which are applied one after the other on the best solution obtained through ABC algorithm. Our first local search repeatedly applies a direction flip based heuristic as long as there is an improvement in solution quality. This heuristic considers each demand one-by-one in their natural order and flips its direction in the solution if doing so reduces the objective function value. It is to be noted that in the case of WRELP problem, flipping the direction of only those demands can possibly reduce the objective

function value which are currently routed through the edge having the maximum load. Similarly, in the case of WRALP problem, flipping the direction of only those demands can possibly reduce the objective function value which are currently routed through the arc having the maximum load. Hence, for efficiency considerations, only such potential demands should be tried for flipping.

Our second local search repeatedly applies a heuristic, which considers a pair of demands at a time, as long as there is an improvement in solution quality. This heuristic considers each demand one-by-one in some random order (for implementing the random ordering, every time the heuristic begins execution, a random sequence of demands is generated) and if the demand under consideration, say i, contributes to the maximum load, then we try to pair it with some other demand so that flipping the directions of both the demands together reduces the objective function value. The demands are tried for pairing with demand i in their natural order. If such a pairing is found then we immediately do the required flipping and then again all demands are tried one-by-one for pairing with i. Only when i can not be paired with any other demand to reduce the objective function value, next demand in the random sequence is considered. Demands in a pair can have the same direction or opposite directions. Again for efficiency considerations, we can curtail the number of demands that need to be tried for pairing with i. Only those demands can possibly pair with i which does not contribute to the maximum load and which have a weight less than that of i. In the case of WRELP, a demand contributes to the maximum load if it is routed through the edge having the maximum load. In the case of WRALP, a demand contributes to the maximum load if it is routed through the arc having the maximum load.

The pseudo-code of hybrid ABC approach is presented in Algorithm 11. In this algorithm, E_n and O_n are the number of employed bees and onlooker bees respectively. *DNS(X)* is a function that determines a new solution X' in the neighborhood of the solution X and returns X' (Section 7.3.5). Pseudo-code for *DNS(X)* is given in Algorithm 10. Function *BTS(e_1, e_2,...,e_E)* implements the binary tournament selection method (Section 7.3.4). This function returns the index of the solution selected. Function *Perturb(X)* perturbs a solution X to generate a new solution X' as per Section 7.3.6. *LS_1(best)* and *LS_2(best)* are two functions that implement the two local searches described in Section 7.3.7. These two local searches are applied once on the best solution returned by ABC algorithm.

Algorithm 11: Pseudo code of hybrid ABC approach

```
Generate E_n initial employed bee solutions e_1, e_2,...,e_{E_n};
best ← Best solution among e_1, e_2, ..., e_{E_n};
while Termination criteria is not met do
    for i ← 1 to E_n do
        e_i' ← DNS(e_i);
        if e_i' is better than e_i then
            e_i ← e_i';
        else if e_i has not improved over last S_it iterations then
            e_i ← Perturb(e_i) ;
        if e_i is better than best then
            best ← e_i;
    for i ← 1 to O_n do
        j ← BTS(e_1, e_2, ..., e_{E_n});
        o_i ← DNS(e_j);
        if o_i is better than e_j then
            e_j ← o_i;
        if o_i is better than best then
            best ← o_i;
best ← LS_1(best);
best ← LS_2(best);
return best;
```

7.4 Computational results

To evaluate the performance of HABC-A and HABC-E, we have used the same 19 instances as used in [160] and [158]. The number of nodes (n) in these instances vary from 5 to 30, and the number of demands (m) in these instances varies from 6 to 435. To generate these instances, six equally spaced values of n in the interval $[5, 30]$ are considered, i.e., $n \in \{5, 10, 15, 20, 25, 30\}$. Hence, with respect to n, there are six cases which are called case 1 ($n = 5$), case 2 ($n = 10$) and so on. The rings with 5/10/15 nodes are characterised as ordinary sized rings and rings with 20/25/30 as extremely large rings. For a n node ring, $\frac{n(n-1)}{2}$ demand pairs are possible. So if the demand

set contains all these pairs then the demand set is said to be complete; otherwise, it is said to be partial. Four different cases are considered with respect to the type of demand set. The first three cases are applicable for all values of n, whereas the last case is applicable to only $n = 30$. The first case consists of complete set of demands, the second case consists of partial set of demands with $max(\lceil \frac{n(n-1)}{4} \rceil, 8)$ demand pairs randomly chosen from complete set of demands and the third case consists of partial set of demands with $max(\lceil \frac{n(n-1)}{8} \rceil, 6)$ demand pairs randomly chosen from complete set of demands. The weights associated with demands is considered to be uniformly distributed in the interval [5, 100] in these three cases. For each combination of n and these three demand cases, a single instance is generated leading to a total of 18 instances. The last demand case is applicable for $n = 30$ only and consists of a complete set of demands with weights uniformly distributed in [1, 500]. Again a single instance is generated for this case, thereby, leading to a grandtotal of 19 instances. These instances have the name of the form Cxy where $x \in \{1, 2, 3, 4, 5, 6\}$ is the case with respect to the value of n and $y \in \{1, 2, 3, 4\}$ is the case with respect to demand set. Please also note that $y = 4$ only when $x = 6$. Except for larger ring sizes and higher variance in demand weights, these instances are similar to the instances used previously in the literature for ring loading problems. Table 7.1 presents the characteristics of these test instances along with their best known values (BKV) for WRALP and WRELP.

Table 7.2 compares the average execution time of HABC-A on WRALP instances with that of genetic algorithm (GA), hybrid differential evolution algorithm (HDEM), hybrid particle swarm optimisation algorithm (HDPSO) and artificial bee colony algorithm (ABC). The first three approaches are from [160], whereas the last approach is from [158]. Data for GA, HDEM, HDPSO and ABC approaches have been taken from [160] where all the approaches were executed on a 2.66 GHz Intel Q9450 processor based system and average time and iterations to reach the best solution were reported for each method. From average time and iteration to reach the best solution, we got average time per iteration and multiplying it with the total number of iterations allotted yielded the execution time. This is done to follow the standard practice of comparing execution times of various approaches. No matter how fast the best solution is obtained, it is returned only when the program finishes execution, and hence, comparison of time required to reach the best solution does not make sense. For small instances C11, C12, C13, C21, C22, C23, C32, C33 and C41, Bernardino et al. [160] did not

Table 7.1: Characteristics of test instances along with their best known values (BKV)

Instance	#Nodes(n)	#Demands(m)	BKV(WRALP)	BKV(WRELP)
C11	5	10	161	185
C12	5	8	116	137
C13	5	6	116	137
C21	10	45	525	583
C22	10	23	243	352
C23	10	12	141	199
C31	15	105	1574	1657
C32	15	50	941	941
C33	15	25	563	618
C41	20	190	2581	2745
C42	20	93	1482	1760
C43	20	40	612	683
C51	25	300	4265	4304
C52	25	150	2323	2488
C53	25	61	912	1015
C61	30	435	5762	5953
C62	30	201	2696	2901
C63	30	92	1453	1506
C64	30	435	27779	29245

report the average time till best precisely and mentioned that they are < 0.001 seconds, and hence, for these instances we have not reported the execution times precisely in Table 7.2. However, precise execution times for HABC-A are reported in Table 7.4. In a manner similar to Table 7.2, the Table 7.3 compares the average execution time of HABC-E on WRELP instances with that of GA, HDEM, HDPSO and ABC approaches. These tables clearly show the superiority of HABC-A and HABC-E in terms of execution times over other methods on large instances of WRALP and WRELP both. Moreover, the difference in speed between our approaches and other methods widens with an increase in the number of demands. Please note that we have executed our approaches on a 2.83 GHz Intel Q9550 processor based system which is different from the system used to execute GA, HDEM, HDPSO and ABC approaches (2.66 GHz Intel

Table 7.2: Execution times of various approaches in seconds on WRALP instances

Instance	GA	HDEM	HDPSO	ABC	HABC-A
C11	<0.10	<0.10	<0.10	<0.10	<0.10
C12	<0.10	<0.10	<0.10	<0.10	<0.10
C13	<0.10	<0.10	<0.10	<0.10	<0.10
C21	<0.10	<0.10	<0.10	<0.10	<0.10
C22	<0.10	<0.10	<0.10	<0.10	<0.10
C23	<0.10	<0.10	<0.10	<0.10	<0.10
C31	0.33	1.00	0.50	0.67	0.23
C32	<0.10	<0.10	<0.10	<0.10	<0.10
C33	<0.10	<0.10	<0.10	<0.10	<0.10
C41	0.60	0.90	0.53	1.20	0.53
C42	0.19	0.30	0.16	0.3	0.22
C43	<0.10	<0.10	<0.10	<0.10	<0.10
C51	4.69	10.00	2.50	5.00	1.15
C52	1.00	2.00	1.00	2.13	0.39
C53	0.10	0.19	0.09	0.19	0.16
C61	20.19	64.29	15.63	28.13	2.03
C62	3.33	7.50	3.00	5.00	0.65
C63	1.25	2.50	1.25	2.50	0.22
C64	5.00	16.67	10.00	10.00	2.99

Q9450 processor). However, Q9550 and Q9450 processors belong to the same series of processors and Q9550 is immediate successor of Q9450. Hence, there is only a slight difference is processing speeds of the two systems and conclusion drawn here takes into account this difference.

To show the relative contribution of two local searches and artificial bee colony framework, we have implemented HABC-A and HABC-E approaches without any local search and with only the first local search. The versions without any local search will be referred to as ABC-A and ABC-E, whereas the versions with only first local search will be referred to as FABC-A and FABC-E respectively. Table 7.4 & Table 7.5 report the results. For each instance, we report the best solution (column Best), average solution quality (column Avg) and standard deviation of solution values (column SD) over 100

Table 7.3: Execution times of various approaches in seconds on WRELP instances

Instance	GA	HDEM	HDPSO	ABC	HABC-A
C11	<0.10	<0.10	<0.10	<0.10	<0.10
C12	<0.10	<0.10	<0.10	<0.10	<0.10
C13	<0.10	<0.10	<0.10	<0.10	<0.10
C21	<0.10	<0.10	<0.10	<0.10	<0.10
C22	<0.10	<0.10	<0.10	<0.10	<0.10
C23	<0.10	<0.10	<0.10	<0.10	<0.10
C31	0.33	0.67	0.50	0.40	0.19
C32	<0.10	<0.10	<0.10	<0.10	<0.10
C33	<0.10	<0.10	<0.10	<0.10	<0.10
C41	0.45	0.75	0.50	0.40	0.45
C42	0.19	0.33	0.20	0.40	0.18
C43	<0.10	<0.10	<0.10	<0.10	<0.10
C51	5.00	6.25	3.33	6.25	1.58
C52	1.00	2.00	0.80	1.33	0.39
C53	0.10	0.25	0.13	0.17	0.13
C61	22.5	60.00	17.50	18.50	2.00
C62	4.29	10.00	5.00	8.33	0.55
C63	1.25	3.75	1.25	2.50	0.19
C64	3.75	12.50	3.13	4.17	2.67

runs found by HABC-A, FABC-A, ABC-A or HABC-E, FABC-E, ABC-E as the case may be along with their respective average execution times (column AvT) in seconds. These tables clearly show that local searches contribute little to the success of artificial bee colony algorithm which is capable of finding best solutions on its own in most runs.

Table 7.4: Relative contribution of two local searches and artificial bee colony framework on WRALP instances

Instance	HABC-A				FABC-A				ABC-A			
	Best	Avg	SD	AvT	Best	Avg	SD	AvT	Best	Avg	SD	AvT
C11	161	161.00	0.00	0.01	161	161.00	0.00	0.01	161	161.00	0.00	0.01
C12	116	116.00	0.00	0.01	116	116.00	0.00	0.01	116	116.00	0.00	0.01
C13	116	116.00	0.00	0.01	116	116.00	0.00	0.01	116	116.00	0.00	0.01
C21	525	525.00	0.00	0.07	525	525.00	0.00	0.07	525	525.00	0.00	0.07
C22	243	243.00	0.00	0.05	243	243.00	0.00	0.05	243	243.09	0.51	0.05
C23	141	141.00	0.00	0.02	141	141.00	0.00	0.02	141	141.00	0.00	0.02
C31	1574	1574.00	0.00	0.23	1574	1574.00	0.00	0.23	1574	1574.00	0.00	0.23
C32	941	941.00	0.00	0.09	941	941.00	0.00	0.09	941	941.00	0.00	0.09
C33	563	563.00	0.00	0.04	563	563.00	0.00	0.04	563	563.00	0.00	0.04
C41	2581	2581.00	0.00	0.53	2581	2581.00	0.00	0.52	2581	2581.00	0.00	0.52
C42	1482	1482.00	0.00	0.22	1482	1482.00	0.00	0.22	1482	1482.00	0.00	0.22
C43	612	612.00	0.00	0.07	612	612.00	0.00	0.07	612	612.00	0.00	0.07
C51	4265	4265.00	0.00	1.15	4265	4265.00	0.00	1.15	4265	4265.00	0.00	1.15
C52	2323	2323.00	0.00	0.39	2323	2323.00	0.00	0.39	2323	2323.00	0.00	0.39
C53	912	912.00	0.00	0.16	912	912.00	0.00	0.16	912	912.00	0.00	0.16
C61	5762	5762.00	0.00	2.03	5762	5762.00	0.00	2.02	5762	5762.00	0.00	2.02
C62	2696	2696.00	0.00	0.65	2696	2696.00	0.00	0.65	2696	2696.00	0.00	0.65
C63	1453	1453.00	0.00	0.22	1453	1453.00	0.00	0.22	1453	1453.00	0.00	0.22
C64	27779	27779.00	0.00	2.99	27779	27779.06	0.28	2.98	27779	27779.23	0.58	2.98

Table 7.5: Relative contribution of two local searches and artificial bee colony framework on WRELP instances

Instance	HABC-E				FABC-E				ABC-E			
	Best	Avg	SD	AvT	Best	Avg	SD	AvT	Best	Avg	SD	AvT
C11	185	185.00	0.00	0.01	185	185.00	0.00	0.01	185	185.00	0.00	0.01
C12	137	137.00	0.00	0.01	137	137.00	0.00	0.01	137	137.00	0.00	0.01
C13	137	137.00	0.00	0.01	137	137.00	0.00	0.01	137	137.00	0.00	0.01
C21	583	583.00	0.00	0.07	583	583.00	0.00	0.07	583	583.00	0.00	0.07
C22	352	352.00	0.00	0.03	352	352.00	0.00	0.03	352	352.00	0.00	0.03
C23	199	199.00	0.00	0.02	199	199.00	0.00	0.02	199	199.00	0.00	0.02
C31	1657	1657.00	0.00	0.19	1657	1657.00	0.00	0.19	1657	1657.00	0.00	0.19
C32	941	941.00	0.00	0.08	941	941.00	0.00	0.08	941	941.00	0.00	0.08
C33	618	618.00	0.00	0.04	618	618.00	0.00	0.04	618	618.00	0.00	0.04
C41	2745	2745.00	0.00	0.45	2745	2745.00	0.00	0.45	2745	2745.00	0.00	0.45
C42	1760	1760.00	0.00	0.18	1760	1760.00	0.00	0.18	1760	1760.00	0.00	0.18
C43	683	683.00	0.00	0.07	683	683.00	0.00	0.07	683	683.00	0.00	0.07
C51	4283	4284.11	1.16	1.58	4283	4284.55	1.24	1.58	4283	4284.59	1.30	1.58
C52	2488	2488.00	0.00	0.39	2488	2488.00	0.00	0.39	2488	2488.00	0.00	0.39
C53	1015	1015.00	0.00	0.13	1015	1015.00	0.00	0.13	1015	1015.00	0.00	0.13
C61	5953	5953.00	0.00	2.00	5953	5953.00	0.00	2.00	5953	5953.00	0.00	2.00
C62	2901	2901.00	0.00	0.55	2901	2901.00	0.00	0.55	2901	2901.00	0.00	0.55
C63	1506	1506.00	0.00	0.19	1506	1506.00	0.00	0.19	1506	1506.00	0.00	0.19
C64	29245	29245.00	0.00	2.67	29245	29245.02	0.14	2.67	29245	29245.06	0.24	2.67

7.5 Conclusions

In this chapter, we have proposed hybrid artificial bee colony algorithm based approaches for two real-world $\mathcal{NP}$-hard problems belonging to the domain of optical ring networks. Computational results on the standard benchmark instances of the problems show the effectiveness of the proposed approaches. Our ABC approaches are capable of finding high quality solutions on their own even without the use of local search.

Chapter 8

Conclusions and directions for future research

Facility location/assignment problems occur in diverse walks of life, and new problems continue to emerge due to several factors such as advancement in technology, demographical changes, change in business strategies and change in perspectives. Hence, providing solutions to such problems is an evergreen field of research. In this thesis, we have studied four $\mathcal{NP}$-hard facility location problems which are based on either fundamental facility location models or their simple extensions. We have also studied three $\mathcal{NP}$-hard facility assignment problems in this thesis. This thesis is focussed on developing ABC algorithm based approaches for all these facility assignment/location problems, and, wherever possible making these approaches perform as good as or better than the state-of-the-art approaches for these problems. An IWO algorithm is also developed with the same intention for one facility location problem. These are the major contributions of the thesis. In addition, we have developed new local search heuristics for some problems.

In the following, we describe the contributions made and the possible directions in which future research can be carried out based on the work reported in various chapters.

In Chapter 2, an ABC algorithm based approach is presented for solving the p-median problem, where the best solution obtained through ABC algorithm is improved further by a local search. The performance of the proposed approach is compared against some state-of-the-art approaches. This comparison is done using two widely used p-median benchmark sets, viz. OR-Library test instances and Galvao test in-

stances. Computational results establish the superiority of our approach in comparison to other approaches.

In comparison to the solution representation scheme used in a previously proposed ABC algorithm [99], in this chapter, we have proposed a better scheme to represent a solution of p-median problem which is free from redundancy and decoding overhead. Our neighboring solution generation operator also make use of the fact that if a facility is located at a particular demand point in one good solution, then the likelihood of a facility being located at the same demand point in several good solutions is high. This is the reason behind randomly choosing another solution and copying facility locations which are common between this randomly chosen solution and solution under consideration to the neighboring solution being generated. This is also the reason for copying the leftover facility locations from this randomly chosen solution to the neighboring solution being generated. Similar neighboring solution generation method can be developed for other facility location problems also. A different local search may be tried to enhance the performance of our approach. We have applied the local search only on the best solution obtained through ABC algorithm. The performance of our proposed approach can be improved further at the cost of an additional computational burden if a local search is applied on every solution generated through ABC algorithm. A possible future work is to extend our ABC approach to capacitated p-median problem where each facility is supposed to have a capacity and each demand point has a demand, and the sumtotal of demands associated with demand points allocated to a particular facility can not exceed the capacity of that facility. Approaches similar to our approach can also be developed for other related facility location problems also.

Chapter 3 extends the ABC approach developed in the previous chapter to the positive/negative weighted p-median problem. A 1-interchange heuristic is applied K times on each neighboring solution generated in the ABC algorithm, where each time a randomly chosen facility location is tried for exchange with a non-facility location. In addition, the same local search as used in the previous chapter is applied at the end of ABC algorithm. However, instead of applying this local search on the best solution obtained through ABC algorithm, it is applied on L best solutions obtained through ABC algorithm. The parameter K and L represent trade-offs between solution quality and computational cost. The performance of our approach has been compared on the standard benchmark instances of the problem with genetic algorithm (GA)

and ant colony optimization (ACO) based approaches, which are two best approaches available in the literature for the positive/negative weighted p-median problem. Our ABC approach improved the best known solution values of over 10% of the instances. The relative performance of different approaches changes depending on the types of instances. However, GA is clearly better in terms of overall solution quality, though the ABC approach is faster than GA. As far as the comparison between ABC and ACO is concerned, the former approach is much better than the latter approach on instances where half of the weights are negative under both the models. However, ABC is slower than ACO.

Chapter 4 is devoted to the solution of cooperative maximum covering location problem (CMCLP) using an ABC algorithm based approach. The performance of the proposed approach has been compared with two interchange based heuristics. These two heuristics are the only heuristic approaches available in the literature for CMCLP problem. Our ABC algorithm outperformed these two heuristics in terms of solution quality, but at the expense of larger execution times.

To generate a neighboring solution, we have deleted a number of facilities as the strategy of deleting a single facility is not able to improve the solution quality much. Once some facilities are deleted, some of the nodes become uncovered. However, as the number of potential location of facilities, which can cover yet to be covered nodes is infinite, so instead of copying facilities from another employed bee solution like in Chapter 2 and Chapter 3, we begin by finding all those points which provide exact coverage for at least one yet to be covered node. In this way, we have reduced the number of potential locations for facilities to a finite set. Obviously, a facility has to be located at one of these points only in a bid to cover as many nodes as possible. Similar strategies to curtail the search space can be designed for other related problems also.

Our ABC algorithm based approach is the first metaheuristic approach for the CMCLP. Hence, it will serve as a baseline approach for any future metaheuristic approach for CMCLP. Averbakh et al. [35] reported that they also experimented with tabu search and variable neighborhood search based approaches. However, they observed that these approaches provide only a marginal improvement over interchange heuristics, and, hence, they did not present results of these metaheuristics in [35]. On the other hand, our ABC approach significantly improves the results obtained through interchange heuristics. Hence, population-based metaheuristics seem more appropriate

for this problem. A possible future work is to develop some other metaheuristic approaches for the CMCLP and compare them with our ABC approach and two heuristics of [35]. Approaches similar to our ABC approach can be developed for other related problems employing cooperative coverage model. For example, our approach can be easily extended to obnoxious cooperative maximum covering location problem (OCMCLP) where a node is uncovered if its cooperative coverage is less than or equal to a given threshold T and the goal is to maximize the total weight of uncovered nodes.

Chapter 5 is focussed on solving p-center problem through two metaheuristic approaches, viz. ABC algorithm and IWO algorithm. In comparison to the best approach available in the literature, viz. BCOi [124], computational results on the standard benchmark instances for this problem demonstrate the superiority of our proposed approaches in terms of solution quality. As far as the comparison between our two proposed approaches is concerned, both the approaches performed quite similar though IWO is slightly better in terms of solution quality, it is slower also.

Two neighboring solution generation methods have been used in this chapter. To generate a neighboring solution, the first method, deletes some centers and then adds an equal number of new centers, whereas the second method adds some new centers first and then delete an equal number of centers. Both the methods utilize the concept of critical distance introduced in [124] while adding nodes. On the other hand, both methods iteratively delete centers where during each iteration, the center whose deletion has minimal adverse effect on objective function is deleted. It is worthwhile to mention that we have tried random addition and/or deletion of centers, but all these strategies performed poorly. Considering the failure of these purely random strategies, and the success of BCOi, ABC and IWO based approaches in solving the p-center problem, any future approach has to utilize the concept of critical distance in one way or the other to achieve comparable or better results. Our approaches have not made use of any local search. Performance of our proposed approaches can be improved further with the help of a suitable local search strategy at the cost of additional computational burden. A possible future work is to consider a bi-objective version of the problem where one objective is same as p-center problem and the second objective is same as p-median problem and study the trade-off between these two objectives.

Chapter 6 is concerned with a facility assignment problem, viz. terminal assignment (TA) problem. To tackle this problem, an ABC algorithm based approach is presented.

Two local searches are also developed to improve the best solution obtained through ABC algorithm, but they improve the solution only marginally. The performance of the proposed approach is compared with four best methods available in the literature on standard benchmark instances of the problem and found to be similar. ABC algorithm provides good quality solutions in lesser execution times for the majority of the instances. Solution encoding schemes different from the one used here may be tried to improve the performance of the ABC algorithm. A possible future work is to solve the terminal assignment problem under a different objective where either no balance function is used or a balance function different from Equation (6.4) is used or a different relative weightage is given to the value of the balance function in the objective function (Equation (6.5)). For example, the balance function defined in Equation (6.4) is based on the number of terminals assigned to a concentrator. A different balance function can consider the ratio of the sum of weights of terminals assigned to a concentrator to the capacity of that concentrator. Another possible future work is to study a bi-objective version of the problem where the first objective is to minimize the sumtotal of distances between the terminals and their assigned concentrators, and, the second objective is to minimize the unbalance in load as measured by a balance function.

Chapter 7 is another chapter devoted to facility assignment problems where two $\mathcal{NP}$-hard problems arising in optical ring networks are solved using ABC algorithm based approaches. The first problem is referred to as weighted ring edge-loading problem (WRELP), whereas the second problem is referred to as weighted ring arc-loading problem (WRALP). We have developed two new local searches for WRELP/WRALP which are applied one-after-the-other on the best solution obtained through ABC Algorithm. We have compared the performance of the proposed approaches with four best approaches available in the literature on the standard benchmark instances of the problem. Two approaches among these four approaches are based on swarm intelligence and include an artificial bee colony algorithm based approach and a particle swarm optimization based approach. Computational results demonstrate the superiority of our proposed approaches over these four approaches. A comparison between the results of our ABC approaches with and without the use of local search shows that our ABC approaches are capable of finding high quality solutions to WRELP/WRALP on their own without the use of local search.

A possible future work is to develop metaheuristic approaches for the multi-objective versions of ring loading problems involving multiple costs. Another possible future work is to consider a related problem where links are assumed to have bandwidth constraints, and, the goal is to maximize either the number of demands or the sum of weights of demands that can be routed while respecting the bandwidth constraints.

This thesis has considered seven problems in all. Out of these seven problems, first four are facility location problems, whereas the last three are facility assignment problems. We have developed ABC algorithm based approaches for all the seven problems. Besides, an IWO algorithm based approach is also developed for one facility location problem. Based on the approaches presented in this thesis, we can conclude that a properly designed ABC algorithm based approach can compete with any other state-of-the-art metaheuristic approach for any facility location/assignment problem. While designing an approach based on ABC algorithm for any problem, solution encoding, and neighboring solution generation method should be given special attention as they are vital to the success of an ABC algorithm. All our approaches make use of problem-specific knowledge which is utilized for one or more of the following – solution encoding, neighboring solution generation, initial solution generation and local search. Actually, without the appropriate use of problem-specific knowledge, no new metaheuristic approach for any problem can compete with the state-of-the-art metaheuristic approaches for that problem as all such approaches rely heavily on problem-specific knowledge to boost their performance.

In the end, it is pertinent to mention that the approaches presented in this thesis do not preclude the possibility of development of even better ABC algorithm based approaches for these problems. All our approaches exploit some features of the problem at hand. Identifying and analyzing such features for each problem and exploiting them either alone or in combination with some of those already in use may lead to further improvements in ABC algorithm and other metaheuristic techniques based approaches for these problems.

References

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

[2] R. Church and C.R. Velle. **The maximal covering location problem**. *Papers in regional science*, **32**(1):101–118, 1974. (2, 5, 6, 60)

[3] V. Marianov and D. Serra. *4 Location problems in the public sector:In facility location:applications and theory*. Springer, 2002. (2)

[4] A.B. Arabani and R.Z. Farahani. **Facility location dynamics: An overview of classifications and applications**. *Computers & industrial engineering*, **62**(1):408–420, 2012. (2)

[5] R.Z. Farahani, M. SteadieSeifi, and N. Asgari. **Multiple criteria facility location problems: A survey**. *Applied mathematical modelling*, **34**(7):1689–1709, 2010. (2)

[6] 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. (2)

[7] C.S. ReVelle and H.A. Eiselt. **Location analysis: a synthesis and survey**. *European journal of operational research*, **165**(1):1–19, 2005. (2)

[8] 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. (2)

[9] 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. (2, 5)

[10] L. Cooper. **Heuristic methods for location-allocation problems**. *Siam review*, **6**(1):37–53, 1964. (2)

[11] S.K. Jacobsen. **Heuristics for the capacitated plant location model**. *European journal of operational research*, **12**(3):253–261, 1983. (2)

[12] 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. (2, 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. (2, 4, 6)

[14] 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. (2)

[15] 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. (2, 7, 23)

[16] 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. (2)

[17] Z. Drezner. *Facility location:A survey of applications and methods*. Springer, 1995. (2, 3, 4)

[18] Z. Drezner and H. W. Hamacher. *Facility location: applications and theory*. Springer science & business media, 2001. (2, 3)

[19] P.B. Mirchandani and R.L. Francis. *Discrete location theory*. Wiley interscience series in discrete mathematics and optimization. Wiley, 1990. (3)

[20] M.S. Daskin. *Network and discrete location: models, algorithms, and applications*. John wiley & sons, 2011. (3)

[21] R.F. Love, J.J.G. Morris, and G.O. Wesolowsky. *Facilities location: models & methods*. Publications in operations research. North-Holland, 1988. (3)

[22] A. Ghosh and G. Rushton. *Spatial analysis and location-allocation models*. Van nostrand reinhold, 1987. (3)

[23] P. Hansen, J. Henderson, M. Labbé, J. Peeters, and J F Thisse. *Systems of cities and facility location*, **4**. Taylor & francis, 2013. (3)

[24] J.F Thisse and H.Zoller. *Locational analysis of public facilities*. Studies in mathematical and managerial economics. Elsevier, 1983. (3)

[25] H.A. Taha. *Operations research: An introduction*. Pearson, 2011. (3)

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

[27] F. Plastria. **Continuous location problems: research, results and questions**. *Facility location: a survey of applications and methods*, pages 85–127, 1995. (4)

[28] 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. (4)

[29] J. Current, M. Daskin, and D. Schilling. **3 Discrete network location models**. *Facility location applications and theory*, pages 81–118, 2004. (4)

[30] 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. (4, 5, 7)

[31] O. Berman, Z. Drezner, and D. Krass. **Cooperative cover location problems: the planar case**. *IIE Transactions*, **42**(3):232–246, 2009. (5, 60, 61, 62)

[32] O. Berman, Z. Drezner, and D. Krass. **Generalized coverage: new developments in covering location models**. *Computers & operations research*, **37**(10):1675–1687, 2010. (5)

[33] F. Plastria. **Continuous covering location problems**. *Facility location: applications and theory*, **1**, 2002. (5)

[34] J. Current and M. ÓKelly. **Locating emergency warning sirens**. *Decision sciences*, **23**(1):221–234, 1992. (5)

[35] I. Averbakh, O. Berman, D. Krass, J. Kalcsics, and S. Nickel. **Cooperative covering problems on networks**. *Networks*, **63**(4):334–349, 2014. (5, 62, 64, 66, 68, 69, 70, 119, 120)

[36] O. Berman, Z. Drezner, and D. Krass. **Discrete cooperative covering problems**. *Journal of the operational research society*, **62**(11):2002–2012, 2011. (5, 62)

[37] O. Berman. **The p maximal cover - p partial center problem on networks**. *European journal of operational research*, **72**(2):432–442, 1994. (5, 62)

[38] P. Cappanera, G. Gallo, and F. Maffioli. **Discrete facility location and routing of obnoxious activities**. *Discrete applied mathematics*, **133**(1):3–28, 2003. (5, 42)

[39] C.-H. Chung. **Recent applications of the maximal covering location planning (MCLP) model**. *Journal of the operational research society*, **37**(8):735–746, 1986. (6)

[40] R. Huang, S. Kim, and M.B.C. Menezes. **Facility location for large-scale emergencies**. *Annals of operations research*, **181**(1):271–286, 2010. (7)

[41] H.W. Kuhn. **The Hungarian method for the assignment problem**. *Naval research logistics quarterly*, **2**(1-2):83–97, 1955. (8)

[42] G Terry Ross and Richard M Soland. **Modeling facility location problems as generalized assignment problems**. *Management science*, **24**(3):345–357, 1977. (8)

[43] S. Sahni and T. Gonzalez. **p-complete approximation problems**. *Journal of the association for computing machinery(JACM)*, **23**(3):555–565, 1976. (8)

[44] G. Beni and J. Wang. **Swarm intelligence in cellular robotic systems**. In *Robots and biological systems: towards a new bionics?*, pages 703–712. Springer, 1993. (11)

[45] E. Bonabeau, M. Dorigo, and G.Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford university press, 1999. (11)

[46] D. Karaboga. **An idea based on honey bee swarm for numerical optimization**. Technical report, Erciyes university, engineering faculty, computer engineering department, 2005. (12, 29)

[47] B. Basturk and D. Karaboga. **A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm**. *Journal of global optimization*, **39**(3):459–471, 2007. (12)

[48] D. Karaboga and B. Basturk. **Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems**. In *Proceedings of the international fuzzy systems association world congress*, pages 789–798. Springer, 2007. (12)

[49] D. Karaboga and B. Basturk. **On the performance of artificial bee colony (ABC) algorithm**. *Applied soft computing*, **8**(1):687–697, 2008. (12)

[50] D. Karaboga and B. Akay. **A modified artificial bee colony (ABC) algorithm for constrained optimization problems**. *Applied soft computing*, **11**(3):3021–3031, 2011. (12)

[51] A. Singh. **An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem**. *Applied soft computing*, **9**(2):625–631, 2009. (12, 26)

[52] Q.-K. Pan, M.F. Tasgetiren, P.N. Suganthan, and T.J. Chua. **A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem**. *Information sciences*, **181**(12):2455–2468, 2011. (12, 88)

[53] S. Sundar and A. Singh. **A swarm intelligence approach to the quadratic multiple knapsack problem**. In *Proceddings of the international conference on neural information processing*, pages 626–633. Springer, 2010. (12)

[54] W.-F. Gao and S.-Y Liu. **Improved artificial bee colony algorithm for global optimization**. *Information processing letters*, **111**(17):871–882, 2011. (12)

[55] W.-F. Gao and S.-Y Liu. **A modified artificial bee colony algorithm**. *Computers & operations research*, **39**(3):687–697, 2012. (12)

[56] B. Akay and D. Karaboga. **A modified artificial bee colony algorithm for real-parameter optimization**. *Information sciences*, **192**:120–142, 2012. (12)

[57] T. K. Sharma and M. Pant. **Enhancing the food locations in an artificial bee colony algorithm**. *Soft computing*, **17**(10):1939–1965, 2013. (12)

[58] T. K. Sharma, M. Pant, and F. Neri. **Changing factor based food sources in artificial bee colony**. In *IEEE symposium on swarm intelligence*, pages 128–135. IEEE, 2014. (12)

[59] C. Ozturk, E. Hancer, and D. Karaboga. **A novel binary artificial bee colony algorithm based on genetic operators**. *Information sciences*, **297**:154–170, 2015. (12)

[60] D.E. Goldberg and K. Deb. **A comparative analysis of selection schemes used in genetic algorithms**. *Foundations of genetic algorithms*, **1**:69–93, 1991. (15, 29)

[61] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga. **A comprehensive survey: artificial bee colony (ABC) algorithm and applications**. *Artificial intelligence review*, **42**(1):21–57, 2014. (15)

[62] A.R. Mehrabian and C. Lucas. **A novel numerical optimization algorithm inspired from weed colonization**. *Ecological informatics*, **1**(4):355–366, 2006. (15, 16)

[63] H.G. Baker. *The genetics of colonizing species*. Springer, 1965. (16)

[64] X. Zhang, Y. Wang, G. Cui, Y. Niu, and J. Xu. **Application of a novel IWO to the design of encoding sequences for DNA computing**. *Computers & mathematics with applications*, **57**(11):2001–2008, 2009. (16)

[65] G.G. Roy, P. Chakroborty., Z. Shi-Zheng, S. Das, and P.N. Suganthan. **Artificial foraging weeds for global numerical optimization over continuous spaces**. In *Proceedings of the IEEE congress on evolutionary computation*, pages 1–8. IEEE, 2010. (16)

[66] S. Roy, S.M. Islam, S. Das, and S. Ghosh. **Multimodal optimization by artificial weed colonies enhanced with localized group search optimizers**. *Applied soft computing*, **13**(1):27–46, 2013. (16)

[67] A. Basak, D. Maity, and S. Das. **A differential invasive weed optimization algorithm for improved global numerical optimization**. *Applied mathematics and computation*, **219**(12):6645–6668, 2013. (16)

[68] D. Kundu, K. Suresh, S. Ghosh, S. Das, B.K. Panigrahi, and S. Das. **Multi-objective optimization with artificial weed colonies**. *Information sciences*, **181**(12):2441–2454, 2011. (16)

[69] A. Dastranj, H. Abiri, and A. Mallahzadeh. **Design of a broadband cosecant squared pattern reflector antenna using IWO algorithm**. *IEEE transactions on antennas and propagation*, **61**(7):3895–3900, 2013. (16)

[70] D.I. Abu-Al-Nadi, O.M.K. Alsmadi, Z.S. Abo-Hammour, M.F. Hawa, and J.S. Rahhal. **Invasive weed optimization for model order reduction of linear MIMO systems**. *Applied mathematical modelling*, **37**(6):4570–4577, 2013. (16)

[71] P. Ramezani, M. Ahangaran, and X.-S. Yang. **Constrained optimisation and robust function optimisation with EIWO**. *International journal of bio-inspired computation*, **5**(2):84–98, 2013. (16)

[72] P. Venkatesh and A. Singh. **Two metaheuristic approaches for the multiple traveling salesperson problem**. *Applied soft computing*, **26**:74–89, 2015. (16, 79)

[73] S. Peng, A.-J. Ouyang, and J.J. Zhang. **An adaptive invasive weed optimization algorithm**. *International journal of pattern recognition and artificial intelligence*, **29**(02), 2015. (16)

[74] H-Y Sang, P-Y Duan, and J-Q Li. **A discrete invasive weed optimization algorithm for the no-wait lot-streaming flow shop scheduling problems**. In *International conference on intelligent computing*, pages 517–526. Springer, 2016. (16)

[75] O. Kariv and S.L. Hakimi. **An algorithmic approach to network location problems. II: The p-medians**. *SIAM journal on applied mathematics*, **37**(3):539–560, 1979. (22)

[76] A.A. Kuehn and M.J. Hamburger. **A heuristic program for locating warehouses**. *Management science*, **9**(4):643–666, 1963. (23)

[77] E. Feldman, F.A. Lehrer, and T.L. Ray. **Warehouse location under continuous economies of scale**. *Management science*, **12**(9):670–684, 1966. (23)

[78] M.B. Teitz and P. Bart. **Heuristic methods for estimating the generalized vertex median of a weighted graph**. *Operations research*, **16**(5):955–961, 1968. (23)

[79] RA. Whitaker. **A fast algorithm for the greedy interchange for large-scale clustering and median location problems**. *Information systems and operational research*, **21**(2):95–108, 1983. (23)

[80] R.D. Galvao. **Use of lagrangean relaxation in the solution of uncapacitated facility location problems**. *Location science*, **1**:57–70, 1993. (23)

[81] J.E. Beasley. **Lagrangean heuristics for location problems**. *European journal of operational research*, **65**(3):383–399, 1993. (23)

[82] P. Avella, M. Boccia, S. Salerno, and I. Vasilyev. **An aggregation heuristic for large scale p-median problem**. *Computers & operations research*, **39**(7):1625–1632, 2012. (23)

[83] F. Chiyoshi and R.D. Galvao. **A statistical analysis of simulated annealing applied to the p-median problem**. *Annals of operations research*, **96**(1-4):61–74, 2000. (23, 30, 32, 33)

[84] E. Rolland, D.A. Schilling, and J.R. Current. **An efficient tabu search procedure for the p-median problem**. *European journal of operational research*, **96**(2):329–342, 1997. (23)

[85] M. Maroszekz and C. Rettig. **A Tabu search for the p-median problem**. *Technical report, institute of information systems, school of business, university of hamburg*, 2008. (23, 30)

[86] P. Hansen and N. Mladenović. **Variable neighborhood search for the p-median**. *Location science*, **5**(4):207–226, 1997. (23)

[87] C.M. Hosage and M.F. Goodchild. **Discrete space location-allocation solutions from genetic algorithms**. *Annals of operations research*, **6**(2):35–46, 1986. (23)

[88] C. Dibble and P.J. Densham. **Generating interesting alternatives in GIS and SDSS using genetic algorithms**. In *Proceedings of the GIS/LIS '93 PROCEEDINGS,*, **1**, pages 180–189. Amer soc photogrammetry & remote sensing, 1993. (23)

[89] B. Bozkaya, J. Zhang, and E. Erkut. **An efficient genetic algorithm for the p-median problem**. *Facility location: applications and theory*, pages 179–205, 2002. (23)

[90] O. Alp, E. Erkut, and Z. Drezner. **An efficient genetic algorithm for the p-median problem**. *Annals of operations research*, **122**(1):21–42, 2003. (23, 30, 32, 33)

[91] P.J. Densham and G. Rushton. **A more efficient heuristic for solving large p-median problems**. *Papers in regional science*, **71**(3):307–329, 1992. (23)

[92] T-Y. Cheung Z. Dai. **A new heuristic approach for the p-median problem**. *Journal of the operational research society*, **48**(9):950–960, 1997. (23)

[93] K.E. Rosing, C.S. ReVelle, and D.A. Schilling. **A gamma heuristic for the p-median problem**. *European journal of operational research*, **117**(3):522–532, 1999. (23)

[94] N. Özçakir and M. Basti. **Particle swarm optimization algorithm approach for solving p-median facility location selection problem**. *Istanbul university faculty of management journal*, **41**:241–257, 2012. (23, 30)

[95] M. Sevkli, R. Mamedsaidov, and F. Camci. **A novel discrete particle swarm optimization for p-median problem**. *Journal of king saud university engineering sciences*, **26**(1):11–19, 2014. (23, 30, 32, 33)

[96] M.G.C. Resende and R. Werneck. **A hybrid heuristic for the p-median problem**. *Journal of heuristics*, **10**(1):59–88, 2004. (23)

[97] M.G.C. Resende and R.F. Werneck. **A fast swap-based local search procedure for location problems**. *Annals of operations research*, **150**(1):205–230, 2007. (23)

[98] J.A.M. Perez, J.L.R. Garcia, and M. Moreno. **A parallel genetic algorithm for the discrete p-median problem**. *Studies in locational analysis*, **7**:131–141, 1994. (23)

[99] M. Basti and M. Sevkli. **An artificial bee colony algorithm for the p-median facility location problem**. *International journal of metaheuristics*, **4**(1):91–113, 2015. (23, 25, 27, 28, 29, 30, 31, 33, 118)

[100] R.D. Galvão and C. ReVelle. **A Lagrangean heuristic for the maximal covering location problem**. *European journal of operational research*, **88**(1):114–123, 1996. (31)

[101] J.J. Dongarra. *Performance of various computers using standard linear equations software*. University of tennessee. computer science department, 1993. (33)

[102] E. Carrizosa and F. Plastria. **Location of semi-obnoxious facilities**. *Studies in locational analysis*, **12**(1999):1–27, 1999. (42)

[103] F. Plastria. **Optimal location of undesirable facilities: an overview**. *Belgian journal of operational research, statistics and computer science*, **36**:109–127, 1996. (42)

[104] R.E. Burkard and J. Krarup. **A linear algorithm for the pos/neg-weighted 1-median problem on a cactus**. *Computing*, **60**(3):193–215, 1998. (42)

[105] R.E. Burkard, E. Çela, and H. Dollani. **2-medians in trees with pos/neg weights**. *Discrete applied mathematics*, **105**(1):51–71, 2000. (42, 43)

[106] R.E. Burkard and J. Fathali. **A polynomial method for the pos/neg weighted 3-median problem on a tree**. *Mathematical methods of operations research*, **65**(2):229–238, 2007. (43)

[107] J. Fathali and H.T. Kakhki. **Solving the p-median problem with pos/neg weights by variable neighborhood search and some results for special cases**. *European journal of operational research*, **170**(2):440–462, 2006. (43)

[108] J. Fathali, H.T. Kakhki, and R.E. Burkard. **An ant colony algorithm for the pos/neg weighted p-median problem**. *Central european journal of operations research*, **14**(3):229–246, 2006. (43, 45, 56)

[109] J. Fathali. **A genetic algorithm for the p-median problem with pos/neg weights**. *Applied mathematics and computation*, **183**(2):1071–1083, 2006. (43, 45, 56)

[110] Z. Drezner. **Noteon a modified one-center model**. *Management science*, **27**(7):848–851, 1981. (62)

[111] C.D.T. Watson-Gandy. **Heuristic procedures for the m-partial cover problem on a plane**. *European journal of operational research*, **11**(2):149–157, 1982. (62)

[112] Z. Drezner. **The p-cover problem**. *European journal of operational research*, **26**(2):312–313, 1986. (62)

[113] C. ReVelle. **The maximum capture or sphere of influence location problem: Hotelling revisited on a network**. *Journal of regional science*, **26**(2):343–358, 1986. (62)

[114] A. Suzuki and Z. Drezner. **The p-center location problem in an area**. *Location science*, **4**(1):69–82, 1996. (62)

[115] Z. Drezner and A. Suzuki. **Covering continuous demand in the plane**. *Journal of the operational research society*, **61**(5):878–881, 2010. (62)

[116] 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. (62)

[117] 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. (71)

[118] D. Chen and R. Chen. **New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems**. *Computers & operations research*, **36**(5):1646–1655, 2009. (71)

[119] Z. Drezner. **The planar two-center and two-median problems**. *Transportation science*, **18**(4):351–361, 1984. (72)

[120] G.Y. Handler. **p-center problems.** In P. Mirchandani and R.L. Francis, editors, *Discrete location theory*, pages 305–315. John wiley & sons, 1990. (72)

[121] N. Mladenović, M. Labbé, and P. Hansen. **Solving the p-Center problem with tabu search and variable neighborhood search**. *Networks*, **42**(1):48–64, 2003. (72, 81)

[122] C. Caruso, A. Colorni, and L. Aloi. **Dominant, an algorithm for the p-center problem**. *European journal of operational research*, **149**(1):53–64, 2003. (72)

[123] J.A. Pacheco and S.Casado. **Solving two location models with few facilities by using a hybrid heuristic: a real health resources case**. *Computers & operations research*, **32**(12):3075–3091, 2005. (72, 81)

[124] 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. (72, 75, 76, 77, 78, 81, 86, 120)

[125] R. Chandrasekaran and A. Tamir. **Polynomially bounded algorithms for locating p-centers on a tree**. *Mathematical programming*, **22**(1):304–315, 1982. (72)

[126] B. Pelegrin. **Heuristic methods for the p-center problem**. *Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle. Locational analysis*, **25**(1):65–72, 1991. (72)

[127] R. Hassin, A. Levin, and D. Morad. **Lexicographic local search and the p-center problem**. *European journal of operational research*, **151**(2):265–279, 2003. (72)

[128] T. Davidović, D. Teodorović, and M. Šelmić. **Bee colony optimization Part I: The algorithm overview**. *Yugoslav journal of operations research*, **25**(1), 2014. (72)

[129] S. Khuri and T. Chiu. **Heuristic algorithms for the terminal assignment problem**. In *Proceedings of the 1997 ACM symposium on applied computing*, pages 247–251. ACM, 1997. (87, 88)

[130] F.N. Abuali, D.A. Schoenefeld, and R.L. Wainwright. **Terminal assignment in a communications network using genetic algorithms**. In *Proceedings of the 22nd annual ACM computer science conference on scaling up : meeting the challenge of complexity in real-world computing applications*, CSC '94, pages 74–81. ACM, 1994. (87, 88)

[131] Aaron A. Kershenbaum. *Telecommunications network design algorithms.* McGraw-hill, Inc., 1993. (87)

[132] S. Salcedo-Sanz and X. Yao. **A hybrid Hopfield network-genetic algorithm approach for the terminal assignment problem**. *IEEE transactions on systems, man, and cybernetics, Part B (cybernetics)*, **34**(6):2343–2353, 2004. (88)

[133] Y. Xu, S. Salcedo-Sanz, and X. Yao. **Non-standard cost terminal assignment problems using tabu search approach**. In *Proceedings of the evolutionary computation, 2004. CEC2004. Congress on*, **2**, pages 2302–2306. IEEE, 2004. (88)

[134] E.M. Bernardino, A.M. Bernardino J.M. Sánchez-Pérez, J.A. Gómez-Pulido, and M.A. Vega-Rodríguez. **Solving the terminal assignment problem using a local search genetic algorithm**. In *Proceedings of the international symposium on distributed computing and artificial intelligence*, pages 225–234. Springer, 2009. (88, 94)

[135] E.M. Bernardino, A.M. Bernardino, and J.M. Sánchez-Pérez. **Tabu search vs hybrid genetic algorithm to solve the terminal assignment problem**. In *Proceedings of the IADIS international conference applied computing*, pages 404–409. IADIS, 2008. (88, 94)

[136] E.M. Bernardino, A.M .Bernardino J.M. Sánchez-Pérez, J.A. Gómez-Pulido, and M.A. Vega-Rodríguez. **A hybrid differential evolution algorithm for solving the terminal assignment problem**. In *Proceedings of the international work-conference on artificial neural networks*, pages 179–186. Springer, 2009. (88)

[137] E.M. Bernardino, A.M. Bernardino J.M. Sánchez-Pérez, J.A. Gómez-Pulido, and M.A. Vega-Rodríguez. **A hybrid DE algorithm with a multiple strategy for solving the terminal assignment problem**. In *Proceedings of the hellenic conference on artificial intelligence*, pages 303–308. Springer, 2010. (88, 94)

[138] E.M. Bernardino, A.M. Bernardino J.M. Sánchez-Pérez, J.A. Gómez-Pulido, and M.A. Vega-Rodríguez. **Discrete differential evolution algorithm for solving the terminal assignment problem**. In *Proceedings of the international conference on parallel problem solving from nature*, pages 229–239. Springer, 2010. (88, 89, 90, 91, 94, 96)

[139] W.J. Goralski. *Sonet*. McGraw-Hill Professional, 2002. (98)

[140] F. Davik, M. Yilmaz, S. Gjessing, and N. Uzun. **IEEE 802.17 resilient packet ring tutorial**. *IEEE communications magazine*, **42**(3):112–118, 2004. (99)

[141] RPR Alliance. **A summary and overview of the IEEE 802.17 Resilient Packet Ring Standard**, 2004. (99)

[142] P. Yuan, V. Gambiroza, and E. Knightly. **The IEEE 802.17 media access protocol for high-speed metropolitan-area resilient packet rings**. *IEEE network*, **18**(3):8–15, 2004. (99, 100)

[143] Y.-S. Myung, H.-G. Kim, and D.-W. Tcha. **Optimal load balancing on SONET bidirectional rings**. *Operations research*, **45**(1):148–152, 1997. (99)

[144] Y.-S. Myung and H.-G. Kim. **On the ring loading problem with demand splitting**. *Operations research letters*, **32**(2):167–173, 2004. (99)

[145] B.-F. Wang. **Linear time algorithms for the ring loading problem with demand splitting**. *Journal of algorithms*, **54**(1):45–57, 2005. (99)

[146] K.S. Cho, U.G. Joo, H.S. Lee, B.T. Kim, and W.D. Lee. **Efficient load balancing algorithms for a resilient packet**. *ETRI journal*, **27**(1):110–113, 2005. (99, 100)

[147] S. Cosares and I. Saniee. **An optimization problem related to balancing loads on SONET rings**. *Telecommunication systems*, **3**(2):165–181, 1994. (100)

[148] A. Schrijver, P. Seymour, and P. Winkler. **The ring loading problem**. *SIAM review*, **41**(4):777–791, 1999. (100)

[149] S. Khanna. **A polynomial time approximation scheme for the sonet ring loading problem**. *Bell labs technical journal*, **2**(2):36–41, 1997. (100)

[150] M. Dell'Amico, M. Labbé, and F. Maffioli. **Exact solution of the SONET ring loading problem**. *Operations research letters*, **25**(3):119–129, 1999. (100)

[151] P. Kubat and J.M. Smith. **Balancing traffic flows in resilient packet rings**. In *Performance evaluation and planning methods for the next generation internet*, pages 125–140. Springer, 2005. (100)

[152] N. Karunanithi and T. Carpenter. **A ring loading application of genetic algorithms**. In *Proceedings of the 1994 ACM symposium on applied computing*, pages 227–231. ACM, 1994. (100)

[153] S.-S. Kim, I.-H. Kim, V. Mani, and H.J. Kim. **Ant colony optimization for SONET ring loading problem**. *International journal of innovative computing, information and control*, **4**(7):1617–1626, 2008. (100)

[154] A.M. Bernardino, E.M. Bernardino, J.M. Sánchez-Pérez, J.A. Gómez-Pulidoua, and M.A. Vega-Rodríguez. **Solving the ring loading problem using genetic algorithms with intelligent multiple operators**. In *Proceedings of the international symposium on distributed computing and artificial intelligence*, pages 235–244. Springer, 2009. (100)

[155] A.M. Bernardino, E.M. Bernardino, J.M. Sánchez-Pérez, J.A. Gómez-Pulidoua, and M.A. Vega-Rodríguez. **Solving the weighted ring edge-loading problem without demand splitting using a hybrid differential evolution algorithm**. In *Proceedings of the IEEE 34th Conference on local computer networks*, pages 562–568. IEEE, 2009. (100)

[156] A.M. Bernardino, E.M. Bernardino, J.M. Sánchez-Pérez, J.A. Gómez-Pulidoua, and M.A. Vega-Rodríguez. **Solving the non-split weighted ring arc-loading problem in a resilient packet ring using particle swarm optimization**. In *Proceedings of the international joint conference on computational intelligence*, pages 230–236. Springer, 2009. (100)

[157] A.M. Bernardino, E.M. Bernardino, J.M. Sánchez-Pérez, J.A. Gómez-Pulidoua, and M.A. Vega-Rodríguez. **A discrete differential evolution algorithm for solving the weighted ring arc loading problem**. In *Proceedings of the international conference on industrial, engineering and other applications of applied intelligent systems*, pages 153–163. Springer, 2010. (100)

[158] A.M. Bernardino, E.M. Bernardino, J.M. Sánchez-Pérez, J.A. Gómez-Pulidoua, and M.A. Vega-Rodríguez. **A hybrid ant colony optimization algorithm for solving the ring arc-loading problem**. In *Proceedings of the hellenic conference on artificial intelligence*, pages 49–59. Springer, 2010. (100, 104, 105, 107, 109, 110)

[159] A.M. Bernardino, E.M. Bernardino, J.M. Sánchez-Pérez, J.A. Gómez-Pulidoua, and M.A. Vega-Rodríguez. **Efficient load balancing for a resilient packet ring using artificial bee colony**. In *Proceedings of the european conference on the applications of evolutionary computation*, pages 61–70. Springer, 2010. (100)

[160] A.M. Bernardino, E.M. Bernardino, J.M. Sánchez-Pérez, J.A. Gómez-Pulidoua, and M.A. Vega-Rodríguez. **Solving ring loading problems using bio-inspired algorithms**. *Journal of network and computer applications*, **34**(2):668–685, 2011. (100, 104, 109, 110)

List of publications

[1] B. Jayalakshmi and A. Singh . **A hybrid artificial bee colony algorithm for the terminal assignment problem** . *In Proceedings of the 5th International Conference on Swarm, Evolutionary and Memetic Computing (SEMCCO)*, LNCS, **8947**, pages 134-144, Springer, 2015.

[2] B. Jayalakshmi and A. Singh . **A hybrid artificial bee colony algorithm for the cooperative maximum covering location problem**. Accepted for publication at *International Journal of Machine Learning and Cybernetics* (DOI:10.1007/s13042-015-0466-y).

[3] B. Jayalakshmi and A. Singh. **A hybrid artificial bee colony algorithm for the p-median problem with positive/negative weights**. Accepted for publication at *OPSEARCH* (DOI:10.1007/s12597-016-0271-8).

[4] B. Jayalakshmi and A. Singh. **A swarm intelligence approach for the p-median problem**. Accepted for publication at *International Journal of Metaheuristics.*

[5] B. Jayalakshmi and A. Singh. **Two swarm intelligence based approaches for the p-center problem**. Communicated to *International Journal of Swarm Intelligence.*

[6] A. Singh and B. Jayalakshmi. **Hybrid artificial bee colony algorithm based approaches for two ring loading problems**. Communicated to *Applied Intelligence.*