Optimizing Infrastructure Placement in Wireless Mesh Networks

A thesis submitted in partial fulfilment

of the requirement for the degree of Doctor of Philosophy

Liqaa Faisal Nawaf

March 2017

Cardiff University
School of Computer Science & Informatics

Declaration

This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date .....

Statement 1

This thesis is being submitted in partial fulfillment of the requirements for the degree of PhD.

Signed ..... (candidate)

Date .....

Statement 2

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references.

Signed ..... (candidate)

Date .....

Statement 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)

Date .....

Copyright © 2017 Liqaa Faisal Nawaf.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

A copy of this document in various transparent and opaque machine-readable formats and related software is available at `http://yourwebsite`.

To my parent
for their patience and support.

Abstract

Wireless Mesh Networks (WMNs) are a promising flexible and low cost technology to efficiently deliver broadband services to communities. In a WMN, a mesh router is deployed at each house, which acts both as a local access point and a relay to other nearby houses. Since mesh routers typically consist of off-the-shelf equipment, the major cost of the network is in the placement and management of Internet Transit Access Points (ITAP) which act as the connection to the internet. In designing a WMN, we therefore aimed to minimize the number of ITAPs required whilst maximizing the traffic that could be served to each house. We investigated heuristic and meta-heuristic approaches with an efficient combination of move operators to solve these placement problems by using single and multi-objective formulations. Many real-world optimisation problems involve dealing with multiple and sometimes conflicting objectives. A multi-objective approach to optimize WMN infrastructure placement design with three conflicting objectives is presented: it aims to minimize the number of ITAPs, maximize the fairness of bandwidth allocation and maximize the coverage to mesh clients. We discuss how such an approach could allow more effective ITAP deployment, enabling a greater number of consumers to obtain internet services. Two approaches are compared during our investigation of multi-objective optimization, namely the weighted sum approach and the use of an evolutionary algorithm. In this thesis we investigate a multi-objective optimization algorithm to solve the WMN infrastructure placement problem. The move operators demonstrate their efficiency when compared to simple Hill Climbing (HC) and Simulated Annealing (SA) for the single objective method.

Acknowledgements

I am heartily thankful to my main supervisor Professor Stuart Allen, for his encouragement, guidance and support from the beginning of my PhD to enable me to develop my research skills and understanding of the subject. Thanks for his great enthusiasm and patience with me. Without his invaluable support and encouragement, this thesis would have not been accomplished. I would like to thank my second supervisor Professor Omer Rana, who has given a great help for my research and for his continued support and knowledgeable guidance. My thanks go to Dr Christine Mumford for her supervision at the start of my PhD.

I would like to thank all members of the School of Computer Science and Informatics for their helpful discussions, comments, feedback, events and facilities. Special thanks to Dr Rob Davies, Mrs Helen Williams and Dr Pamela Munn for their administrative support. A special thanks to Student Support Centre and especially Paula Barker for her support and kindness. She has been invaluable in helping and supporting me through my PhD and I am very grateful.

All the way through my study at Cardiff University, I have been supported and inspired, through interaction with many friends and colleagues within the School of Computer Science. I would like to extend my thanks to Dr Hana Aldahawi for being supportive and a thoughtful friend all through my study. I am deeply grateful to Fatma Alrayes for her support and kindness. A special thanks to my friend Eman Alghamdi for her encouragement. I am grateful to my friends Soha Mohamed, Aseela Al Harthi, Ash-

wan Abdulmunem, Wafa Alorainy, Dr Haya Almagwashi and Dr Shada Alsalamah for their support and friendship, also grateful to all my colleagues and friends in Cardiff University who have always provided encouragement. Thanks go to the Alan & Cyril Body Trust for their support.

It is an honour for me to extend my thanks to my family for their continuous understanding and encouragement especially my parents and Rama. Their love and support and guidance have sustained me throughout my time at Cardiff University.

Contents

List of Figures

List of Tables

List of Algorithms

List of Notations

ACO Ant Colony Optimization

ADSL Asymmetric Digital Subscriber Line

b_h Bandwidth allocated to each house

Cap_e Edge Capacity

Cap_h House Capacity

Cap_i ITAP Capacity

DS Dataset

EA Evolutionary algorithm

f_D Unsatisfied Demand solution

f_I Number of ITAPs solution

f_U Unfairness solution

GA Genetic Algorithm

HC Hill Climbing

h_c House Capacity

IC ITAP capacity

ITAP Internet Transit Access Point

LP Linear Programming

MOEA Multi-objective Evolutionary Algorithm

MOO Multi-Objective Optimization

M Number of Houses

N Number of ITAP location

NSGA-II Non-Dominated Sorting Genetic Algorithm

P Population

P_m Mutation Rate

P_r Probability Rate

PSO Particle Swarm Optimization

QoS Quality of service

SA Simulated Annealing

VoIP Voice over Internet Protocol

W_D Unserved Demand Weight

w_h House Demand

W_I ITAP Weight

WLC Wireless link capacity

WMN Wireless Mesh Network

WRC Wireless range Connectivity

W_U Unfairness Weight

$X_{e,h}$ Flow from edge to house

y_i Number of ITAPs installed at location i

Chapter 1

Introduction

Broadband access has rapidly become a central part of modern life, providing great social and financial benefits. Due to the emergence of areas such as cloud computing, the Internet of Things, online gaming and music/video streaming, broadband traffic in 2018 is expected to be three times that in 2013, possibly rising to a 45-to-80-fold increase by 2030 [1]. Although broadband provision has increased significantly in the UK, this has largely taken place in areas of high population density, due to the high cost of providing wired connections in rural regions.

The cost and difficulty of installing wired connections in rural areas and in developing countries was one of the motivations behind the development of Wireless Mesh Networks (WMNs). WMNs are a flexible solution in which customers cooperate to share their connections to provide wider coverage, and offer many advantages for providing broadband access. The key features of WNMs are their low-cost flexible deployment, ease of expansion and resilience.

A WMN is created through the installation and connection of a wireless *mesh router* at each network user's premises. Each mesh router acts as a standard access point to provide WiFi coverage to local devices, but also acts as part of the network infrastructure, aggregating and forwarding traffic to other customers towards an Internet gateway. In this sense, each network user generates traffic, but is also a relay, forwarding data from other nearby customers to the next node towards the gateway. Internet Transit Access Points (ITAPs) serve as gateways or bridges to the Internet, for example,

through connection to a wired backbone network. The ITAP is the only core network infrastructure required; it shares its Internet connection wirelessly with all the houses in its vicinity in term of wireless range connectivity. An ITAP could use identical hardware to a mesh router (but with a wired connection to the internet), but could in some cases have improved capabilities (e.g. wireless range). To initiate a network a WMN needs a minimum of one ITAP, which can then grow automatically as more clients join. However, although this can lead to network coverage over a large area, further ITAPs will need to be added as the number of clients and thus traffic generated grows. Large or small networks can be formed in this way to serve small urban communities, or millions of residents in a city. The low initial costs make WMN particularly suitable for rural areas or developing countries, avoiding the high cost associated with providing a wired links to every house. In this thesis, we address a number of optimisation problems relating to the placement of ITAPs. These problems involve minimizing the numbers of ITAPs that are required to service the needs of a community. This situation is easily modelled using graph theory. The focus of our work is to demonstrate the effectiveness of novel move operators using a simple ideal link model for problem formulations with single and multi-objectives. A move operator is the transition process from one solution to another within its neighbourhood. A move operator is the transition process from one solution to another within its neighbourhood. Appropriate move operators for meta-heuristic approaches are key to balancing the speed of convergence and diversity in the search space. The model connection depends on distance (or range) and not on interference (due to geographical terrain or the related signal reflection/absorption models). The basic ideal link model is taken from [2]. It shows that by using a suitable combination of move operators to place the ITAPs, rapid improvements can be made to the underlying WMN.

1.1 Research Problem and Motivation

One of the fundamental issues in placing infrastructure to support WMN is the trade-off between the number of ITAPs deployed (i.e. cost) and the level of traffic that can be supported. Generally, for a given distribution of houses and set of potential ITAP locations, the number of ITAPs required to be deployed (to meet traffic demand) is unknown. To illustrate this trade-off, consider, for example, the simple "point to multi point" network shown in Figure 1.1, where two houses each introduce D units of traffic into the network, and each wireless link can support the transfer of B units of traffic. In this network, each house communicates directly with the ITAP; hence the entire capacity of a link can be used to deliver traffic, giving $D = B$. However, note that this requires each house to be in the wireless range of the ITAP, which can lead to a high numbers of ITAPs required. Consider the alternative distribution in Figure 1.2. Although we still have $D = B$, two ITAPs are now required, since $H1$ is outside of the wireless range of the location ITAP2 and $H2$ is outside of the wireless range of the location ITAP1. By moving to the mesh architecture shown in Figure 1.3, we reduce the requirement to use only one ITAP, since house $H1$ can relay traffic to and from $H2$ and the ITAP. However, since the link to the ITAP now needs to accommodate $2D$ units, the traffic that can be delivered to each house drops to $B/2$. This simple case shows the advantages and disadvantages of a mesh architecture compared to a point to multi point network.

To see the effect of ITAP placement, consider a network with potential wireless links as shown in Figure 1.4.

Figure 1.5 shows the traffic on each link if an ITAP is deployed at the location ITAP1, whereas Figure 1.6 shows the traffic distribution if ITAP2 is used. In Figure 1.5, each house can be provided with $B/2$ units of traffic, whereas the network in Figure 1.6 can deliver only $B/4$ units. Note that even in this simple example, the choice of ITAP location has a great effect on the traffic that can be served in the network, motivating our study of an algorithm that automatically deploys ITAPs at different locations.

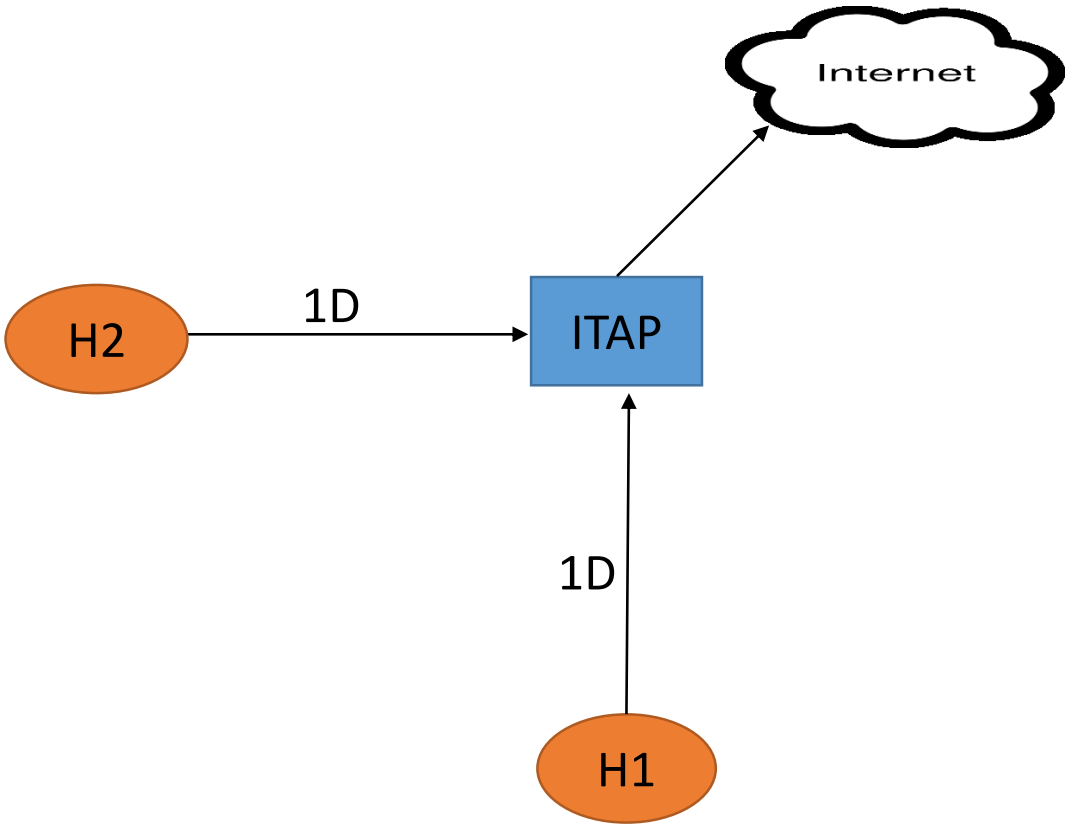

Figure 1.1: Simple network model of two houses and one ITAP for $D \leq B$

In Figure 1.1 the demand of both houses will be satisfied as long as $D \leq B$ with single ITAP.

Figure 1.2: Network model outside the wireless range of the ITAP location for $D \leq B$.

In Figure 1.2 the demand of both houses will be satisfied but needs two ITAPs.

Figure 1.3: Network with in mesh architecture model for $D \leq B/2$

In Figure 1.3 the demand of both houses is greater than B, therefore, each house will receive $B/2$ only. If $D \leq B/2$ then the demand of both houses will be satisfied.

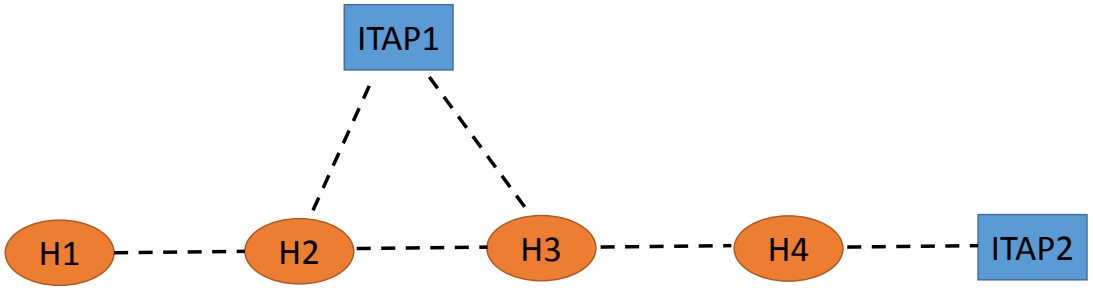

Figure 1.4: Network with potential wireless links

The aim is to develop algorithms capable of selecting the best locations at which to place infrastructure for the WMNs to optimise the multiple objectives of cost, throughput and fairness.

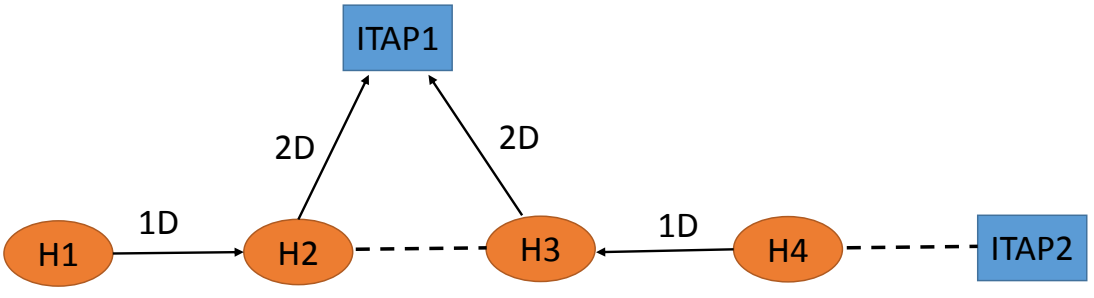

Figure 1.5: Traffic on links towards ITAP1

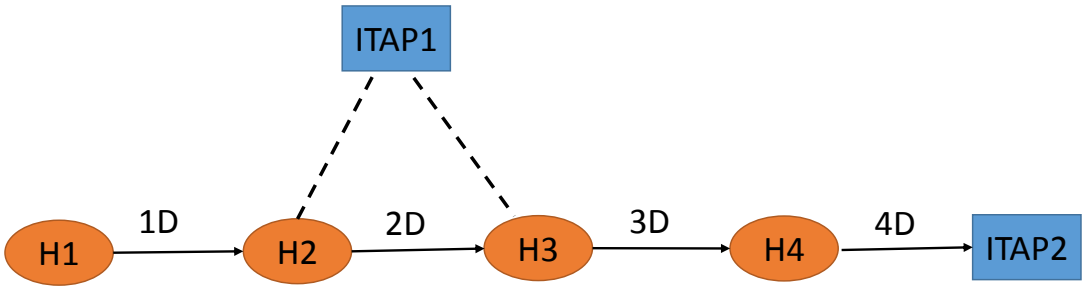

Figure 1.6: Traffic on links towards ITAP2

1.2 Research Aims and Objectives

- Design and implement algorithms for improving the performance and efficiency of WMNs. In terms of challenge, WMN brings the technology to rural areas and developed countries where standard broadband is impracticable.
- Investigate the trade-off between different performance objectives and develop an algorithm that makes use of this trade-off. The novelty of our submission is centred on the implementation of move operators and the effect of move operators that also strike a balance between the objectives.

1.3 Research Hypothesis

This research investigates two hypothesis;

- A single objective approach, using algorithms such as hill climbing and simulated annealing, can effectively design WMNs by using a suitable combination of move operators to place the ITAPs to improve performance and efficiency. This technique provides a better service such as minimising the number of ITAPs and maximising throughput in a WMN.
- With suitable crossover and mutation operators, a multi-objective genetic algorithm approach is more effective than a weighted sum approach in producing

a range of networks that encapsulate the trade-off between the number of ITAPs, throughput and fairness of bandwidth allocation in a WMN.

1.4 Research Contributions

The main contribution of this research is the optimization of ITAP placement in WMN that makes most effective use of different performance metrics (in a combined manner). More specifically we will be:

- Generating simple data sets for the WMN ITAP placement problem.
- Solving the single objective problem of ITAP placement in an efficient way to yield excellent solutions by effective combination of move operators using heuristic and metaheuristic algorithms.
- Demonstrating the importance of move operators through the use of metaheuristics for multi-objective optimization. The evaluation shows the efficiency of using a combination of move operators and presents a better solution for the placement of ITAPs in WMNs.
- Proposing and evaluating aggressive neighbourhood move operators to improve the efficiency of the optimization algorithms.
- Defining and demonstrating the use of move operators for metaheuristics, crossover and mutation operators for a multi-objective Genetic Algorithm.

1.5 Thesis Outline

Chapter 2 provides background and a literature review of related approaches. A detailed explanation of some well-known heuristic and meta-heuristic algorithms

is provided, including Greedy Algorithms, Hill Climbing (HC), Simulated Annealing (SA), Genetic Algorithms (GAs), Ant Colony Optimisation (ACO), and Particle Swarm Optimisation (PSO) algorithms, and approaches to address multi-objective optimization problems.

Chapter 3 investigates of the WMN ITAP placement optimization problem by introducing the problem formulation and investigates a range of benchmark problems. It introduces neighbourhood move operators as part of Hill Climbing and Simulated Annealing algorithms for the ITAP placement problem.

Chapter 4 presents the WMN bandwidth optimization problem and introduces neighbourhood move operators as part of the Hill Climbing and Simulated Annealing algorithms for the ITAP placement problem.

Chapter 5 introduces WMN infrastructure placement as a multi-objective optimisation problem, with the aim of minimizing the number of ITAPs required, maximizing users' demand that can be satisfied, and maximizing the fairness of bandwidth allocation. It shows that the modelling is improved by using a weighted sum method.

Chapter 6 continues the investigation of the WMN ITAP placement optimization problem. Different crossover and mutation operators for a multi-objective Genetic Algorithm are defined. Experimenting with different forms of parameters and applying aggressive parameters.

Chapter 7 summarizes the research undertaken in this thesis and elaborates on the thesis contribution before suggesting future research directions.

List of Publications

The following publications have been produced on the basis of the research presented in this thesis:

- L. Nawaf, C. Mumford, and S. Allen. "Optimizing the Placement of ITAPs in Wireless Mesh Networks by Implementing HC and SA Algorithms."International Conference on Ad Hoc Networks. Springer International Publishing, 2015.
- L. Nawaf, S.M. Allen, and O. Rana, "Internet Transit Access Point placement and bandwidth allocation in Wireless Mesh Networks." In Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual (pp. 1-8). IEEE.

Chapter 2

Background and Literature Review

There is a wealth of literature covering many aspects of wireless mesh networking (WMN), including routing, scheduling, propagation and modelling. In this chapter we restrict our attention to the areas relevant to infrastructure placement.

2.1 ITAP Placement Overview

The installation management cost of ITAPs in WMNs is significant, so it is crucial to minimize the number of ITAPs required whilst maintaining an acceptable Quality of Service (QoS). ITAP placement is addressed in [3], identifying a minimum number of ITAPs so that the QoS requirements are satisfied. In a WMN, traffic is aggregated and forwarded to the ITAPs to perform efficiently. The authors in [3] formulate the ITAP placement problem as an integer linear program (ILP) and show that the problem of finding a minimum number of ITAPs is NP-Hard. In practice, a linear programming (LP) solver such as CPLEX can handle only small networks (of up to 10 ITAPs) under the proposed model of [3], due to the rapid increase in the number of variables and constraints as the network size increases. The authors propose an algorithm that produces recursive approximations of the problem. The approach reduced the number of ITAPs by up to 50% of the number in other algorithms, whilst at the same time exhibiting smooth and consistent performance when subject to different QoS constraints. From [3] it is clear, however, that CPLEX is too slow to use on large networks, and some

initial experimental work undertaken as part of our study also supports this conclusion. In [4] the fundamental issues are the placement of mesh routers in a local network that has one ITAP, and the difficulty of finding the smallest number of mesh routers to satisfy the network coverage, connectivity constraints and meet Internet traffic demand. The degree of increase of the traffic demand with respect to the number of ITAPs increases with the network size, which indicates that in larger networks traffic density has a significant impact on the number of required ITAPs. In [2] the authors developed algorithms to place ITAPs in multi-hop wireless networks, so as to minimize the number of ITAPs while satisfying users' bandwidth requirements. Here the ITAP placement problem is formulated as a linear program and solved via several greedy-based approximation algorithms. In [5] an ITAP deployment problem is described: the ITAP is an essential hop for distributing broadband service and is more expensive than a mesh router which delivers transmit functionality. Thus the number and the location of ITAPs in the architecture determine the performance obtainable and its cost. In [6] the authors observe that the transmission rate and channel utilization required to satisfy the WMNs demand depend to some extent on the device technologies, but in particular on the distance between the mesh client and the ITAPs to which it is connected. Hence, the allocation mechanism influences the number of mesh clients who have a chance to exploit the available bandwidth. The authors in [7] minimize the network installation cost while providing full coverage to wireless mesh clients. In this type of network architecture, a limited number of ITAPs wirelessly connected to the wired network can provide low cost internet connectivity to a large number of mesh routers. The authors propose relaxation-based heuristic algorithms to obtain very good solutions in reasonable computation time. The authors in [7] confirm the proposed optimization approach, demonstrated on a generated synthetic instance of WMNs which, by changing several network parameters, have been solved to optimality using CPLEX. With the proposed heuristics, the numerical results show that the models are able to capture the effect on the network configuration of all relevant parameters such as capacity, communication range and wireless link capacity, providing a promising framework for the planning of

WMNs. The IBM ILOG CPLEX Optimizer tool, CPLEX optimizer [8] can be used to solve some formulations of the ITAP placement problem. Typically CPLEX is used to assess the size of problem that can be solved exactly before alternative approaches are proposed for the general case. CPLEX is used to reveal the size and runtime of an exact solution to a problem. We investigated a different approach, similar to the one proposed in [2], above. The result obtained shows the limitation of runtime in solving large problems using CPLEX; therefore heuristic and meta-heuristic algorithms were used to solve large complex WMN optimization problem of maintaining QoS with a small number of ITAPs. The authors of [7] and [9] propose an optimization approach which indicates that the heuristics can provide close to optimal solutions, even for large instances. The obtained numerical results of [10] show that directional antennas can greatly enhance the performance of WMNs, thus allowing high throughput services. The main advantages of using directional antennas with wireless multi-hop networks are the reduced interference and the possibility of having parallel transmissions among neighbours with a consequent increase in the spatial reuse of radio resources. The authors in [11] indicate that network performance is heavily impacted by wireless interference and congestion, causing considerable frame losses and longer delays. In [12] it is noted that interference can occur among neighbouring mesh routers, especially in urban areas or emergency environments with a dense deployment of WMN equipment, when the coverage areas of the mesh routers overlap. In Chapter 1 the ITAP placement problem was described. The quality of the deployment of WMN depends on two parameters; the traffic demand in each house and the link capacity if the number and position of ITAPs were given. In [7], as a result, the high data demand with low average link rate has an impact on throughput and may lead to unfeasibility. Hence, one of the main problems facing WMN is the reduction of capacity due to the interference caused by simultaneous transmissions. Consuming multiple channels and wireless multi-hop networks can alleviate but not eliminate interference. The performance problems occur for many reasons, such as collision, multi-path interference, traffic slowdown due to congestion, severe co-channel interference due to poor network

planning or to the poor configuration of ITAPs. A well-planned network was therefore sought, since wireless network performance degradation can be traced back to the original design assumptions. The authors in [11] believe that a well-planned and optimized wireless network can often provide extra capacity with the same infrastructure cost; for instance, this may result from more efficient use of radio frequencies. Another way to achieve better network performance is to optimize the placement and characteristics of the ITAPs before network deployment. A careful placement of ITAPs may lead to less congestion, lower delay and eventually better throughput if the distances and the link capacity are taken into account. Focusing on the neighbourhood search for heuristic algorithms, the authors in [13] consider neighbourhood search-based methods to be more powerful than ad hoc methods for achieving near optimal placement of mesh router nodes. The ad hoc methods for the placement of mesh routers are simple methods that explore a range of possible placement topologies. The authors' experimental evaluation demonstrates excellent performance from a swap-based movement neighbourhood search. In [14] the neighbourhoods are based on reversing segments of routes (sub-routes) and exchanging segments between routes. The authors demonstrate that their Variable Neighbourhood Search is very competitive compared with state-of-the-art heuristic algorithms. In [6] the authors address the problem of placing mesh router nodes in WMNs, and consider three different types of movement (Random, Radius and Swap). They find that combining movements produces better performance than individual movements. They also show that the Simulated Annealing algorithm clearly outperforms the Hill Climbing algorithm. The above literature review suggests that a combination of different move operators would address the network optimization problem (move operators are discussed in Chapters 3 and 4). In [15], the authors propose the grid-based ITAP (called "gateways") method of deployment using cross-layer throughput optimization to evaluate the maximal flow of different gateway deployment schemes in random WMNs. The maximal flow is solved by linear programming. For the cross-layer optimization, the flow supported by mesh networks not only needs to satisfy the capacity constraints, but also needs to be schedulable by all links without in-

terference. The authors demonstrate that the position of the gateways in WMNs affects the total network throughput and the throughput achieved by this method is better than either random or fixed deployment methods. The major difference between their work and ours is that their goal is to maximize the throughput with a fixed number of ITAPs, while ours is to minimize the number of ITAPs deployed and maximize the throughput with fair bandwidth allocation. Chapter 3 investigates both grid based and uniform deployment for comparison. In spite of all the benefits and differences, WMNs routers are usually built on the basis of a similar hardware platform; consequently, algorithms, such as heuristic and meta-heuristic methods, can be developed to determine the best locations at which to place the infrastructure for WMNs to optimize the multiple objectives that arise. Since different ITAP placements lead to different mesh backbone topology and affects the network throughput, it is important to find the optimal ITAP placement to maximize the throughput.

2.2 Overview of Throughput and Fair Bandwidth Allocation

The network throughput is the rate of successful data transferred through a communication channel per unit of time, and the end-to-end throughput increases in step with the link capacity. Bandwidth is the raw capability of a communications channel to move data; in other words, bandwidth can be defined as the maximum amount of data that can move from one point to another over a given amount of time. Correspondingly, the wireless link state i.e. (bandwidth, delay, packet loss) between the nodes affects the QoS, i.e. a good link state guarantees the optimal QoS. In the network, the throughput improves in direct proportion to the number of ITAPS.

In [16] the problem discussed is the capacity of WMNs in relation to the fairness of multi-hop networks; for example, as more nodes are installed, the reliability and network coverage increase. If one or more nodes fail, the packets will be re-routed around

the failed node; similarly if one ITAP fails, the others will take over its traffic. From [16] it is clear that the mesh structure guarantees the accessibility of multiple paths for each node in the network, because originally each node was connected to several other nodes and if one drops out of the network, its neighbours simply find another route - therefore, the ability to configure routes is dynamic.

Maintaining fairness in WMNs is important, since, due to forwarding, nodes far away from the ITAPs significantly reduce the overall throughput. However, this has been given less attention than certain other aspects of the task, such as capacity maximization and maintaining connectivity. The optimization goals of [17] are to identify the trade-off between network throughput and fairness, and prevent unfair bandwidth allocation to any user. The authors consider a max-min fairness model which leads to high throughput solutions which guarantee to have the maximum minimum bandwidth allocation value. To enhance fairness, the authors in [18] propose a 2-fold solution. First, to use a queue for every flow that passes through a certain node; and second, to ensure a higher attempt rate for nodes that have to forward more traffic. To this end, they propose to assign a Contention Window to the attempt rate of each node; that is, a function of the total number of flows passing through this node. In [19] the nodes at greater distance (more hops) from the ITAP suffer from bandwidth starvation. Therefore, they propose a transmission scheduler algorithm (TSA) that offers greater fairness and maximizes bandwidth use. The algorithm assigns each node a weight based on its location in the WMN and on its aggregated traffic load. The weight is later used to compute the node transmission time. The results show that the TSA outperforms the default scheduler in terms of the throughput, fairness, overall network throughput and average end-to-end delay of the individual nodes. The authors of [20] analytically verified and modelled the presence of starvation of nodes located two hops away from the ITAP (called the “gateway”) in a linear topology. They demonstrate that the starvation problem can be tackled by setting the contention window of the nodes one hop away from the ITAP to a much higher value than other nodes. The authors in [21] introduce a Fair Access Rate (FAR) method which, to achieve fairness, takes into account the

number of hops and the aggregate traffic. A variable transmission rate is assigned by FAR to relay nodes based on their aggregation level. When the traffic is sufficiently large and the nodes have data to transmit. FAR performs well in terms of both the fairness index and throughput. The throughput falls considerably, however, as the number of hops increases. This is mainly due to traffic aggregation and the saturation tree build-up to the ITAP. In [22] the spatial contention within a wireless channel and the reuse of the channel bandwidth introduces a conflict between optimizing the aggregate allocated bandwidth and achieving fairness. Each flow in the network is given a fair allocation of capacity, no matter how much more contention is perceived than in other flows; the researchers use a construct similar to a conflict graph, called a flow contention graph, to capture interference in the wireless networks. In [12] node clustering and subcarrier allocation are applied to improve system throughput and then allocate bandwidth to facilitate QoS provisioning by standard means of maximum frequency reuse and effective interference control. However, the subcarrier allocation refers to splitting up the radio spectrum into subsections and these are the subcarriers.

The above literature review of methods of fair allocation suggests a formulation of the fair bandwidth allocation problem. Generally, maximizing the throughput may cause bandwidth starvation for some wireless mesh nodes, particularly nodes far from the ITAP, which have a more significant effect on congestion within the network - indicating the need to consider a multi-objective optimization (MOO) strategy. In order to achieve a good bandwidth allocation with regard to both fairness and throughput, a formulation of a fair bandwidth allocation model is proposed that achieves greater fairness and maximizes the available bandwidth; this aspect is described in more detail in Chapter 5.

In [23] the cost effective broadband provision problem uses a consumer device (mesh router) to transmit data through the network. The authors use this technique to investigate in detail the effect of the relationship between consumer demand and subscription price on the design of optimized mesh networks. In a WMN, service coverage

depends on the clients' capacity to route traffic via subscribed nodes to the point-of-presence (POP); an effective network placement depends on an adequate density of subscribers and the placement of devoted seed nodes to ease routing. A seed node is a mesh router placed as infrastructure to facilitate coverage. The model is flexible and extensible enough to suggest other scenarios, such as requests for extra seed nodes, which may be significant in improving QoS as well as offering extra coverage. [24] demonstrates how the performance of WMN deployments can be improved through optimized personalized subscription pricing. The introduced model and optimization algorithm can determine an individual price for each potential subscriber that reflects their importance to the provision of connectivity within the network. Demand curves help to model the consumer's decisions that relate to their willingness to pay the price of the service offered to them. Here the optimization of pricing also works successfully, as demonstrated, when the model for willingness to pay is extended to include measures of QoS. In [25] a multi commodity Minimum-Cost Maximum-Flow algorithm for routing multiple unicast traffic flows in WMNs was introduced. The routing algorithm is claimed to attain the maximum flow of multiple unicast commodities while simultaneously attaining minimum cost. The author proposes an iterative solution algorithm called Successive Relaxation, where the sub graphs originally contain minimum-distance paths and where selective sub graphs are extended so that longer-distance paths are included when appropriate. The suggested routing algorithm eliminates undesirable edges from consideration, creating a significantly smaller LP to solve, with significantly faster solutions. The authors in [26] present LP formulations of the constrained *Max-Flow-Min-Cost* routing algorithms. Each trunk is represented as a unicast commodity between 2 data centres, which is constrained to flow over a set of feasible edges. In [27] a technique is presented for computing the limited fair capacity of ITAPs as a function of the contention at each ITAP. To achieve a maximized capacity on the current network, the authors propose two online ITAP placement algorithms that use local search operations. The idea of local search is to carefully select a set of ITAPs to close and open a new set of ITAPs, subject to capacity and budget constraints, in or-

der to minimize the objective, i.e. the average hop count, usually reducing hop count and increasing capacity. The experimental results show that the local search algorithms outperform a greedy algorithm by up to 64% and achieve close to the optimal capacity, which supports the use of local search operations on near optimal placements. From the literature review, it can be seen that local search algorithms with suitable heuristic move operators can produce acceptable results in instances that are too large for solutions by exact methods. A two-phase approach is proposed for solving the network optimization problem, in which the underlying framework iterates between the two phases. The first phase focusses the improving of the QoS, as described in Chapter 3. The second phase then attempts to reduce the number of ITAPs required, whilst still maintaining adequate QoS, as described in Chapter 4. To effectively support the different goals of the two phases, a mix of move operators is required.

2.3 Heuristic and Meta-heuristic Algorithms

Heuristic algorithms suggest some approximations to solve the optimization problems but do not guarantee that the best solution will be identified. The goal of the heuristic algorithm is to find as good a solution as possible for all instances of the problem in reasonable time. There are general heuristic strategies such as greedy algorithm that have been successfully applied to various problems. Alternatively, an important subclass of heuristics comprises meta-heuristic algorithms, which are general purpose algorithms that can be applied to a wide range of optimization problems, with only minor modifications to the basic algorithm definition. Many meta-heuristic techniques attempt to mimic biological, physical and natural phenomena. Some heuristic and meta-heuristic algorithms have been applied to solving network optimization problems, as described in the following subsections.

2.3.1 Greedy Algorithm

Greedy Algorithms follow the problem solving heuristic of making the locally optimal choice at each stage in the hope of finding a global optimum, with a decision strategy such that any choice made at any stage takes no consideration of any future state. In [28] the strategy is to make the choice which has the greatest short-term effect on the long-term goal. In some problems this may not produce the optimum solution, i.e. it does not guarantee a globally optimal solution. Greedy algorithms have been widely used in previous studies and have the advantage of being extremely fast compared to other optimization methods such as hill climbing and simulated annealing algorithms, whilst producing reasonably efficient solutions. Greedy Algorithms are recommended in cases where there are many local optima, run time is important and finding a global optimum is not necessary.

2.3.2 Local Search Algorithms

Within the context of searching for a good quality solution, the term **search space** is used to denote all solutions. **A neighbourhood** of solutions is a subset of the search space, generated by making a small change in the current solution. A good quality solution can be found in a class of heuristic and meta-heuristic algorithms that are often called **Local Search**, which moves from one location to another. In Local Search, the new solution is usually generated within the *neighbourhood* of the previous solution. The neighbourhood has one or more **local optima**, denoting the best solutions in this neighbourhood. In general, any local search algorithm starts with an initial solution and then continually tries to find better solutions by searching the neighbourhood of the current solution.

2.3.2.1 Hill Climbing Algorithm (HC)

Hill climbing is the simplest form of local search, being an improvement heuristic that is adopted to enhance feasible solutions through a search mechanism. It starts with a sub-optimal solution to a problem (that is, starting at the base of a hill) and then attempts to repeatedly improve the solution by small steps (neighbourhood moves) until some condition is maximized (the top of the hill is reached). The HC Algorithm will simply accept all neighbouring solutions that are better than the current solution; when HC cannot find any better neighbours, it stops. HC rarely finds a global optimum; it more often gets stuck in a local optimum. Algorithm 2.1 shows the steps of an HC procedure applied to a maximisation problem. HC is frequently applied within other meta-heuristic techniques to improve solution quality at several phases of the search. In this research, HC is used between certain approaches for solving the network optimization problem, as described in Chapters 3 and 4. $N(s)$ represent the neighbourhood function.

Algorithm 2.1: Generic Hill Climbing Algorithm (s, iterations)

```
1  Number of iterations = 0
2  Generate an initial solution s randomly
3  Select s' ∈ N(s) such that f(s') is maximized ;
4  do
5  |  Δ ← f(s') − f(s)
6  |  if Δ > 0 then
7  |  |  s ← s'
8  |  end
9  while Δ > 0;
10 Return s
```

2.3.2.2 The Simulated Annealing Algorithm (SA)

Simulated annealing is a well-known and very powerful meta-heuristic search method and has been shown to be an important tool in a variety of disciplines. It generally gives a "good" solution and guarantees a statistically optimal solution that they can be derived from maximum-likelihood or minimum variance principles for arbitrary problems more than other optimization techniques can claim. SA can deal with arbitrary systems and *cost functions*. It is relatively easy to code and implement, even for complex problems. The main advantages of SA over other local search methods are its flexibility and its ability to approach global optimality. SA is similar to HC, but a little more sophisticated. It has been used successfully in solving many combinatorial optimization problems, and is better at avoiding local optima than HC if it is well implemented. SA was originally inspired by the slow cooling of metals to form crystalline structures of minimum energy, and the Metropolis algorithm [29] first introduced these principles into numerical minimization. The Metropolis algorithm in statistical procedure offers an overview of iterative enhancement, where controlled uphill moves (moves that do not reduce the energy of the system) are probabilistically accepted in the search to obtain a better group and escape local optima. Like HC, SA is launched with a starting configuration, s_0, and then works through a large number of neighbourhood moves (s to s') in an attempt to produce better solutions. Unlike HC, however, the acceptance criterion for s' is less strict, allowing the algorithm to jump out of local optima. If the new solution, s', is better than the current solution, s, it is accepted unconditionally. If, however, the new solution is worse, then it is accepted with a certain probability, related first to how much worse it is; and second, to how high the current "temperature" of the system is. At high temperatures, the system is more likely to accept solutions that are worse. In practice, SA implementation is typically constructed within two nested loops. In the outer loop, the temperature is reduced gradually, and within the inner loop many perturbations of s to s' are tried. At each step, s' faces an acceptance test, based on Δ = Cost(s') - Cost(s), where Δ is the change in cost between

the new and the current solution. The new solution, s', is accepted with a probability of 1, if Cost(s') < Cost(s) (as in HC), and with a probability of $e^{-K\Delta/T}$. Otherwise, where T is the current temperature and K is the **Boltzmann constant**, the probability decreases exponentially.

The annealing temperature is preferably high at the start so that the probability of acceptance will also be high and nearly all new solutions are accepted. The temperature is then gradually reduced. Consequently, the probability of acceptance of low quality solutions reduces and the algorithm acts more or less like hill climbing, i.e., high temperatures allow a better investigation of the search space, while lower temperatures allow fine changes to a good solution. The procedure is repeated until the temperature reaches zero or no further enhancement can be attained. This is equivalent to the atoms of the solid reaching a crystallized state. Implementing SA for a given optimization problem requires choices regarding both general SA parameters and problem specific decisions. The choice of SA parameters is crucial to the performance of the algorithm. These parameters are the initial temperature value T and a temperature function that regulates how the temperature will be modified gradually over time. The temperature update function is typically a proportional temperature function; $T(t+1) = \alpha T(t)$, where t is the current iteration. In the system, the initial temperature is set, $T_0 = 2$, with the temperature reduction coefficients of $alpha = 0.85$ and the constant $K = 3$. To reduce the temperature gradually, these values were used to provide very small reductions of temperature, which led to the slow cooling of the substance until the temperature reached zero. Applying SA also requires a set of problem specific decisions. They consist of recognizing the set of feasible solutions to the problem, outlining a clear objective function, generating an initial solution and outlining a neighbourhood operator that creates moves using the current solution.

The general SA algorithm is defined in Algorithm 2.2, below, where the original optimization problem is assumed to be a minimization function.

SA has shown itself to be extremely promising for portfolio optimization and asset

Algorithm 2.2: Simulated Annealing (s, T, α, max iterations, K)

```
Initialize T
Generate random configuration s
while T > T_min do
    while number of iteration < max iterations do
        Generate new configuration s' within the neighbourhood of s # (s' N(s))
        Δ = Cost(s') − Cost(s)
        if Δ < 0 then
            Cost(s) = Cost(s')
        end
        else
            P = Random (0, 1) {generate at random number in the interval (0, 1)}
            if P ≤ e^(−KΔ/T) then
                Cost(s) = Cost(s')
            end
        end
    end
    Increment number of iterations +1
end
Return s
```

allocation problems, making SA an attractive option for optimization problems. Unfortunately, there is a clear trade-off between the quality of the solutions and the time required to compute it, which is very slow, especially if the cost function is expensive to compute. However, because it produces excellent solutions, it was decided to make use of SA in the research. SA was applied to maximize the throughput, as described in Chapter 3. It was also applied to minimize the number of ITAPs required whilst maintaining an adequate quality of service (QoS), using a single objective method, as outlined in Chapter 4.

2.3.3 Genetic Algorithms (GAs)

The understanding of simulating biological evolution and the choice of natural organisms began in the 1950s. Again nature provides the basis technique for processing information that are both intelligent and multipurpose. Genetic Algorithms are an optimization tool initially established by John Holland in 1975, with its Adaptation in Natural and Artificial Systems MIT Press (2015) [30]. After this, GAs became popular as an intelligent optimization technique that could be used for solving a wide diverse range of hard problems. GAs are a search and optimization technique that owes much to the fundamentals of natural genetics. Some essential procedures are borrowed from genetics and used artificially to structure search algorithms that are solid and require minimum information of the problem. GAs have been regularly implemented, in recent years to solve optimization problems in various domains. GA is a collective term describing a family of stochastic heuristic algorithms based on the natural selection principle of survival of the fittest. The main idea behind these algorithms is to keep evolving a population of candidate solutions one generation after another in the expectation of finding a global optimum or in the worst case a suboptimal solution. Since GAs are population-based algorithms, several solutions can be kept simultaneously throughout the entire process. In GAs, each individual is generally denoted by a string of bits or vectors of integers/real numbers, analogous to *chromosomes* and *genes*, i.e., the parameters of the problem are the genes that are merged together in a solution chromosome. A chromosome is a set of parameters which define a potential solution to the problem that the genetic algorithm is attempting to solve; this set of solutions is known as the population. Every gene is located in a chromosome, and every factor in inheritance is due to a particular gene. Genes specify the structure of a particular inheritance from parents. A fitness value is assigned to each individual, to evaluate its ability to survive and breed. Being selected for reproduction gives highly fit individuals the chance to breed through selecting the “best fit” individuals, good features are spread throughout the population over numerous generations, and the most promising

areas of the search space are discovered. Finally, the population ought to converge towards an optimal or near optimal solution. Convergence means that the population develops in the direction of increasing consistency, and the average population fitness approaches very close to the highest fitness. During the reproduction phase of GA, two important operations have to be performed. The first is called crossover, which allows the basic genetics of parents to pass to their children, their genes to be combined and the next generation to be formed. The second operation is mutation, which is applied to each child produced from a crossover; with a certain small probability of mutation, each gene may be transformed. Hence, crossover allows a rapid exploration of the search space by generating great jumps, while mutation maintains diversity through small amount of random search. However, it is important to note that a number of crossover and mutation operators exist, which implement the same basic principle. In the present research, different crossover and mutation operators were investigated to show the influence of genetic operators on the performance of GA for designing WMN, as explained in more detail in Chapter 6. The basic outline of GA is defined in the following algorithm, Algorithm 2.3.

In the last few years, GAs have shown themselves very well suited to solving larger knapsack problems and general integer programming problems [31]. GA can be viewed as a general purpose search method and optimization method based on the principle of biological evolution [32]. In [33], to optimize nonlinear and multivariable systems, GA with neural network are presented and used extensively. [31] and [32] can be summarised as saying that GAs represent an intelligent search process, since they operate on a population of solutions in promising areas of the search space. Among evolutionary algorithms GA has received great attention [33]. Using the operations of selection, crossover, and mutation, GAs can rapidly scope fit individuals (not always reach the best fit), but are often good enough as solutions to large-scale problems. Since crossover is considered the main GA operator, having to join two solutions rather than one, makes designing an effective crossover operator more complex than defining a mutation operator or a simple neighbourhood move. This generally makes

Algorithm 2.3: Genetic Algorithm: Initialise, Population size, crossover and mutation rate

1 **Coding:** initialise and generate a random population of solutions;
2 **repeat**
3 **Fitness Assignment:** decode and evaluate the fitness of each chromosome;
4 **Selection:** select some chromosomes from the current population for reproduction, where the selection criterion is based on the fitness of the selected parents;
5 **Recombination:** with some probability apply crossover between the selected parents to produce new children;
6 **Mutation:** apply mutation with a small probability to some genes of the newly produced offspring, or to selected members of the population;
7 **Replacement:** integrate the newly generated offspring with the old population to create a new generation;
8 **until** *certain stopping condition is satisfied*;

GA implementation harder than that of SA or simple HC. This research applies a GA to solve the problem of optimizing WMN infrastructure placement, with the aim of minimizing the number of ITAPs required, maximizing the users' demand that can be satisfied and maximising the fairness of bandwidth allocation, as described in more detail in Chapter 6.

2.4 Other Meta-heuristic Algorithms

Other algorithms inspired by nature includes Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) while these are not considered in this thesis, we give a brief overview for compleness. Other authors have applied these approaches in multi-objective optimization problems.

2.4.1 Ant Colony Optimization Algorithm (ACO)

The Ant Colony Optimization algorithm is a meta-heuristic technique that is stimulated by real ant behaviour in food searches based on the generation of a pheromone trail. The core concepts behind this approach were established by Dorigo et al. in 1991 [34]. When an ant travels through paths, from nest to food source, a substance called a 'pheromone' drips from it. A pheromone is a chemical substance produced by the ant that attracts a response in other ants. The other ants choose their paths according to the pheromone concentration, i.e. they choose the paths with the greatest pheromone concentration, because these are the shortest ways to the food. Thus, after a period of time, the greatest number of ants are directed to use the shortest path. This type of indirect communication mediated by pheromone laying is known as 'stigmergy' [35], in which the theory of *positive feedback* is exploited to find the best potential path, based on the choices of previous ants. An optimization algorithm can be established from such ant behavior. The Ant System [36] was the first ACO algorithm, and later, other implementations of the algorithm were developed [37, 38]. The authors in [39] proposed an ITAP placement algorithm based on the ACO algorithm. Constructed on such a model, an evaluation of the numerical results shows that the suggested algorithm elicits much better performance than other schemes. It is also verified to be a cost-effective solution. In [37] the ACO algorithm does not converge so fast and its result is a little worse than Particle Swarm Optimization (PSO). Algorithm 2.4 shows the main steps of the ACO meta-heuristic [28].

2.4.2 Particle Swarm Optimization Algorithm (PSO)

The Particle Swarm Optimization is an evolutionary optimization technique that varies from other evolutionary approaches, such as genetic algorithms, in which there are no crossover and mutation operators and the whole population of particles is sustained throughout the search or exploration operation. PSO is an extremely simple algorithm

Algorithm 2.4: Ant Colony Optimization Algorithm

```
1 Set parameters, initialize pheromone trails
2 while termination condition not met do
3 |   Construct Ant Solutions
4 |   Apply Local Search (optional)
5 |   Update Pheromones
6 end
```

that seems to be effective for optimizing a wide range of functions and was developed by [39, 40]. It models the dynamic movement or behavior of the particles in a search space, while attempting to mimic the design of flowing motion in swarms of birds and exploring the notion of 'collective intelligence' in biological populations. Algorithm 2.5 shows the generic PSO Algorithm [41].

In [42], the authors propose improved particle swarm optimization hybridized with an ant colony (PSACO) approach, for optimizing multimodal continuous functions. The proposed method applies PSO for global optimization and uses the idea of the ant colony approach to update the positions of particles to discover the feasible solution space quickly. The performance study demonstrates the effectiveness and efficiency of the proposed PSACO approach and verified approaches to solve complex optimization problems as successful. Generally, noting the hybridized approach of two effective algorithms can give good results. The ACO algorithm does not converge very fast [37] and the PSO algorithm has no crossover and mutation operators. Hence the GA approach is applied to the ITAP placement problem because it is commonly used.

2.5 Multi-Objective Optimization Approaches

Multi-Objective Optimization (MOO) deals with the task of simultaneously optimizing two or more conflicting objectives with respect to a set of certain constraints. Problems

Algorithm 2.5: Particle Swarm Optimization Algorithm

```
for each particle do
    Initialize particle
end
do
    for each particle do
        Calculate data fitness value;
        if the fitness value is better than the best fitness value (pBest) then
            Set pBest = current fitness value
        end
    end
    #Choose the particle with the best fitness value of all particles as the gBest
    if pBest is better than new gBest then
        Set gBest = pBest
    end
    for each particle do
        Calculate particle velocity according to previous equations;
        Update particle position according to previous equations;
    end
while maximum iterations or minimum error criteria is false;
```

outside the academic world often show an attempt to improve one objective creating difficulties for others. Each multi-objective problem has a set of Pareto-optimal solutions. The solution is called Pareto optimal if none of the objective functions can be improved without worsening other objective values. Each solution represents a different trade-off between the objectives that is said to be "non-dominated" that there is no better solution than the current one in some objective functions, since it is not possible to improve the chance of meeting one criterion without worsening the chance of -meeting another. The goal of the MOO algorithms is to find Pareto-optimal solutions which are

non-dominated solutions for the entire feasible space. In [43] two approaches to solving MOO problems are described. The first is the "weighted-sum method" that uses a single solution updated in every iteration, using a deterministic transition rule for the most part. This method provides only one solution per simulation run. The second consists of "Evolutionary Algorithms" (EAs), mimicking evolutionary principles to drive its search to finding a population of optimal solutions in a single simulation run. The nature of the problem lies in the decision making process, which includes several decision variables, and optimizing a number of objective functions. These objectives may conflict with one another, i.e., an increase of the value of an objective can be obtained only by reducing the value of the others. For example, increasing the coverage by increasing the number of ITAPs deployed and this will increase the cost. The challenge lies in becoming able to search for attractive points in the simultaneous optimization of all the objectives. The solution of the problem will be the best trade-offs between these multiple objectives.

In [44], "Multi Objectivization" is described; a single-objective optimization problem is decomposed into several subcomponents for the sake of a multi-objective approach. This procedure has been found helpful for avoiding local optima in a problem and has attracted significant attention in network optimisation. In [45] multiple objective network planning is considered, where overall interference level is also minimized along with low cost deployment and increased throughput. An efficient population-based search algorithm is proposed to solve this problem. In the proposed scenario, the chosen objectives involve minimizing the number of ITAPs, maximizing throughput and maximizing the fair bandwidth allocation. The two best-known multi-objective approaches are briefly explained below.

2.6 Weighted-sum Approach

The weighted-sum method is one of the most popular approaches used in MOO, due to its simplicity. It combines all of the objectives into a single scalar value converting the MOO problem into a single objective problem and was introduced by Zadeh in [46]. The most general treatments of MOO simply outline the weighted-sum approach. In [43] and [47], the technique scalarizes a set of objectives by multiplying each objective with a user-defined weight. This method is applied to solve an allocation problem with two or more objective functions. Usually, the objectives have different units (i.e., *discrete and real numbers*) with different numerical ranges, making it difficult to choose appropriate weights to control the relative contribution of each objective to the weighted total. The formulation of the objective function can be recognized as a sum of the weighed normalized objectives, which converts the problem into a single-objective optimization problem. In particular, the importance of the weights is not thoroughly explored, and thus, despite the presence of many algorithms for determining weight values, no essential guidelines have been presented for selecting weights for the accurate a priori articulation of preferences. The success of this approach depends on creating suitable selections for the weights. In [48], the method is applied to topology optimization, minimizing compliance and maximizing the first mode of the natural frequency, for two-dimensional plane stress problems. Again, the weights are altered to yield a representation of the Pareto-optimal set. In [49] a weighted sum is used to combine two objective functions in an optimization-based approach to predicting robotic motion. In [50] a three-objective problem is presented, but the weighted sum method is used as a platform for studying various function-transformation methods and their effect on the depiction of the Pareto-optimal set. It is found that a convex combination of functions is advantageous for depicting the Pareto-optimal set, as opposed to determining a single solution. A mixed optimization problem (minimum and maximum) needs to convert all the objectives into one type. It needs next to normalize the objectives, so that each one naturally yields values between 0 and 1. Then the normalized

objective values should be multiplied by the appropriate weight. After the objectives are normalized, a merged objective function $f(x)$ can be designed by summing the weighted normalized objectives, then converting the result to a single objective optimization problem, as shown below in Equation 2.1, reproduced from [47].

$$\text{Minimize } f(x) = \sum_{i=1}^{M} wi \times fi(x) \tag{2.1}$$

where the weight of the ith objective function is normalised to fall in the range [0, 1] and the weights are chosen in such a way that their sum is equal to 1, as shown in Equation 2.2

$$\sum_{i=1}^{M} wi = 1 \tag{2.2}$$

Selecting the most applicable weights for the several essential design objectives needs careful thought and depends on the precise significance of the individual design objectives.

On the basis of this literature review, for the simplicity and ease of use of the weighted-sum approach multiple objective techniques are considered for solving the network optimization problem, where the overall number of ITAPs is minimized simultaneously with high throughput and maximum fairness, through a weighted-sum approach based on SA. In the allocation problem, a MOO approach is required to allow a decision maker to evaluate the different ways of satisfying the objectives. These techniques are discussed further in Chapter 5.

2.6.1 Evolutionary Algorithm Approach

Evolutionary algorithms (EA) are popular approaches which generate Pareto-optimal solutions for MOO problems. They are very powerful techniques, used to find solutions in many authentic search and optimization problems. Many of these problems

have multiple objectives, which raise the need to obtain a set of optimal solutions, known as effective solutions. It has been found that using EA is a highly effective way of finding multiple effective solutions (a set of Pareto-optimal solutions) in a single simulation run. The authors of [51] consider this the main benefit of Multi-objective Evolutionary Algorithms (MOEAs). Population-based approaches such as GAs can easily be extended to solve MOO problems; in this case they are called Multi-objective Evolutionary Algorithms (MOEA). GAs are playing an increasingly important role in MOEAs. GAs have been shown to be capable of handling MOO problems, where different solutions based on the notion of non-dominance can be found. A solution is called non-dominated if it is not dominated by any other solution. In this framework, GAs are called MOEAs. The authors of [52] proposed a multi-objective evolutionary QoS routing algorithm based on GA optimization, called a Multi-constrained Genetic Algorithm for QoS routing (MGAQ). The new options of fitness based functions on result density were proposed to increase the ability to generate feasible routes. The Pareto front is a set of non-dominated solutions, selected as optimal, if no objective can be improved without sacrificing at least one other objective. The principal disadvantage of MGAQ is that some MOEA's are more efficient at finding the Pareto front, because its algorithm has an elite preserving operator, which preserves most of the best chromosomes in a population for the next generation. Additionally it is not clear how the writers in question implemented the algorithm and used only the bandwidth restriction.

In [53] the authors proposed a method for solving routing problems by considering the QoS in WMNs, using the multi-objective approach relying on EAs. In this study, the Non-dominated Sorting based Genetic Algorithm-II (NSGA-II) was invoked for finding different alternative routes that guaranteed the QoS requirements in both the Voice over Internet Protocol (VoIP) and file transfer. The results demonstrate the optimal route found by NSGA-II. The NSGA was one of the first such EAs, proposed by Deb et al. in [54]. Over time an improved version of NSGA was introduced, called NSGA-II, enhancing the convergence and the spread of the solutions. Researchers in

[53] and [55] suggest and support the NSGA-II and show that it has been able to come closer than other EAs to the true Pareto front. For a highly effective way of finding a set of effective solutions, multi-objective approaches may be considered for solving the WMN optimization problem where the overall number of ITAPs is minimized laterally with high throughput and maximum fairness. This is done through proposing a MOEA, specifically the NSGA II, based on the non-dominated sort of individual, as discussed in Chapter 6.

2.7 Chapter Summary

This chapter summarised previous studies of WMN optimization problems and meta-heuristic algorithms. An important category of the ITAP placement and bandwidth allocation literature in the area of WMN was reviewed. These problems are gaining more consideration every day, due to the continuous requirement for optimized ITAP placement in WMN. In the literature review [28, 29, 31, 32, 34, 35, 36, 37, 38, 39, 40] different heuristic and meta-heuristic algorithms can be observed, applied to solving WMN optimization problems. Some of these algorithms can be implemented to show their effectiveness in the WMN optimization problem. From the literature review, it was understood that heuristic and meta-heuristic techniques perform well in most practical situations, which have become increasingly common for researchers in the optimization field. Some exact algorithms were briefly highlighted, which for limited problem sizes can be used to solve problems to optimality. For other practical applications, however, a heuristic or a meta-heuristic approach is generally the favoured option. Some important meta-heuristic techniques were reviewed in this chapter, such as the Hill Climbing, Simulated Annealing, Genetic Algorithms, Ant Colony Optimization and Particle Swarm Optimization Algorithms. A multi-objective optimization approaches were introduced as well.

Chapter 3

Optimization of ITAP Placement to Maximise Throughput

Designing a wireless mesh network (WMN) is a fundamental issue in improving network efficiency and maximising objectives such as coverage and throughput capacity. In this chapter, the WMN ITAP placement problem in multi hop WMNs under certain constraints is addressed. Given a fixed number of ITAPs to be deployed in the network, the question is where to place the ITAPs in the mesh network such that the total throughput is maximised. The interface configuration determines the node throughput capacity while different locations of ITAPs lead to different network topologies and architectures [4]. Since all Internet traffic must pass through one of the ITAPs in the WMN, the deployment of ITAPs for WMNs is critical in determining network performance. The rest of this chapter is organized as follows; Section 3.1 defines the network model, section 3.2 presents the details of the proposed integer linear programming formulations, while section 3.3 calculates the objective function. Section 3.4 specifies the data sets used for experimentation. Section 3.5 briefly highlights some heuristic and meta-heuristic techniques that have been applied to the placement of ITAPs, and section 3.6 defines neighbourhood move operators. Section 3.7 presents the implementation of Hill Climbing (HC) and Simulated Annealing (SA) algorithms in an ideal link model. Section 3.8 describes the proposed network architecture and ITAPs placement in uniform and grid placements. Section 3.9 reports the experimental methodology and results of the algorithm tested, and finally, section 3.10 concludes this chapter.

3.1 Network model

The ideal link model proposed in [3] is used with the aim of deploying a fixed number of ITAPs, while satisfying users' bandwidth demands. Following [2], a network is formed by a set of houses $H = \{h_1, ...h_M\}$, along with a set *(I)* of *N* locations at which ITAPs can be installed. Each node has a location (x, y). Each house *h* has a traffic demand, w_h, and it may be said that a house is served if all traffic at this location can be successfully transmitted to an active ITAP (possibly through a sequence of hops). It is assumed that the traffic from each house can be subdivided and routed along multiple paths simultaneously; hence a maximum flow algorithm is run to compute the satisfied demand under an ideal link model. The connection in this model only depends on distance range not on interference. The link is considered to be binary: either there is a link or there is none. A directed graph *G* is constructed with $H \cup I$ as nodes, with edges joining each pair of nodes that are within wireless range. The wireless range is based on the distance of communication between the nodes. The capacity of each edge $e \in E\ (G)$, Cap_e, is the data rate that can be sustained on this link, and each node has a capacity Cap_h which denotes the ability of a house to process and forward data. Each ITAP also has a capacity limit, based on its connection to the Internet and its processing speed, denoted Cap_i.

3.2 Integer Linear Programming Formulation

Following [2] the model is formally described and its variables and constraints defined. For each edge *e* and house *h*, a variable $x_{e,h}$ is defined to indicate the amount of flow which originated from *h* to the ITAPs that are routed through *e*. For each ITAP *i*, a variable y_i indicates the number of ITAPs installed at the location *i*. The overall objective is then to maximise the throughput with a fixed number of ITAPs. In Formulation 3.1, constraint 3.1 ensures flow conservation, namely, for every house except the house where the flow originated, the total amount of flow entering the house is equal to the

total amount of flow exiting it. The constraint in Equation 3.2 indicates that a house does not receive the flow sent by itself. Constraints (3.3 - 3.5) of the integer program capture the capacity constraints on the edges, houses and ITAPs. Equation 3.6 says that no house is allowed to send any traffic to an ITAP location unless there is sufficient capacity from the ITAPs installed there. This inequality of Equation 3.7 constrains flows to be positive, and follows from the ITAP capacity constraint and the assumption that is an integer in Equation 3.8. Equation 3.9 indicates the fixed number, Y of ITAPs to be deployed.

Formulation 3.1

Maximise $\sum_h \sum_i \sum_{e=(v,i)} x_{e,h}$

Subject to:

$$\sum\nolimits_{e=(v,h')} x_{e,h} = \sum\nolimits_{e=(h',v)} x_{e,h} \qquad \forall h, h' \in H, h' \neq h \tag{3.1}$$

$$\sum\nolimits_{e=(v,h)} x_{e,h} = 0 \qquad \forall h \in H \tag{3.2}$$

$$\sum\nolimits_{h} x_{e,h} \leq Cap_e \qquad \forall e \in E(G) \tag{3.3}$$

$$\sum\nolimits_{h',e=(v,h)} x_{e,h'} \leq Cap_h \qquad \forall h \in H \tag{3.4}$$

$$\sum\nolimits_{h',e=(v,i)} x_{e,h'} \leq Cap_i y_i \qquad \forall i \in I \tag{3.5}$$

$$\sum\nolimits_{e=(v,i)} x_{e,h} \leq w_h y_i \qquad \forall i \in I, h \in H \tag{3.6}$$

$$x_{e,h} \geq 0 \qquad \forall e \in E(G), h \in H \tag{3.7}$$

$$y_i \in \{0, 1, 2, ...\} \qquad \forall i \in I \tag{3.8}$$

$$\sum\nolimits_{i \in x} yi = Y \qquad \forall i \in I \tag{3.9}$$

3.3 Defining the Objective Function

In practice, Formulation 3.1 cannot be solved by an exact IP solver such as CPLEX, since it is too slow to use when the data set is large. To enable meta-heuristic algorithms

to be applied, a flow algorithm is suggested to calculate the objective function for a given solution. Algorithms are applied to place a set of ITAPs in different locations, and the network flow algorithm of Ford-Fulkerson [56] is then used to ensure that demand is satisfied for a specific WMN configuration. The Ford-Fulkerson Algorithm can be used to compute the maximum flow capacity of the edges in the network flow, and hence whether not or a given WMN configuration can support the given user demand. The network graph must be modified before applying a flow algorithm as in [2], it is necessary to complete the graph G_p, by adding a source and a sink and to set the link capacity of these edges to infinity. Each node has an in and out node as virtual nodes; these virtual nodes were added to permit the addition of the capacity on their edges to represent their forwarding capacity, as shown in Figure 3.1. Let the flow be $f(y_1, ..., y_N) = \sum_h \sum_i \sum_{e=(v,i)} x_{e,h}$ where y_i is the number of ITAPs to be installed at location i.

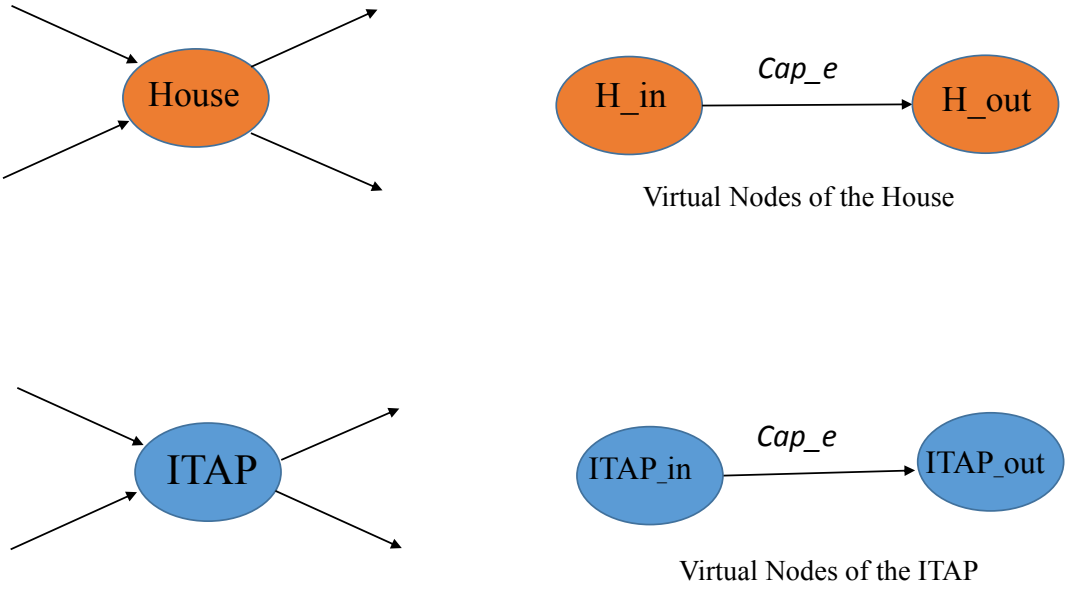

Figure 3.1: Virtual nodes of the House and ITAP with the edge capacity

Every house node is directly connected to the source node and every ITAP node is directly connected to the sink node. Let graph G_p be derived from graph G to calculate the flow. G_p has vertices $H_{in} \cup H_{out} \cup ITAP_{in} \cup ITAP_{out}$. The following definition

of the whole sets is as follows:

1. House h node:
    - For each $h \in H$ two vertices are created, H_{in} and H_{out}.
    - Add edge between H_{in} and H_{out} vertices with Capacity Cap_h.
    - Each edge in the graph between h and ITAP gives vertices between H_{out} and $ITAP_{in}$.
2. Edge (u, v)
    - For each edge $e = (u, v) \in E(G)$ add an edge u_{out}, v_{in} to G_p, between the corresponding in and out vertices.
    - The new edge has capacity Cap_e
3. $ITAP_i$ node
    - For each $i \in I$ two vertices are created, $ITAP_{in}$ and $ITAP_{out}$.
    - Add edge between $ITAP_{in}$ and $ITAP_{out}$ vertices with Capacity Cap_i.
    - Sink is connected to each $ITAP_{out}$.

The connection of edges for the graph G_p would start from the source node and go through H_{in}, H_{out}, $ITAP_{in}$ and $ITAP_{out}$ and then to the sink node, to compute the maximum flow of the network, as shown below in Figure 3.2.

3.4 Data sets

A simple algorithm was created to generate simple data sets to evaluate the solutions in this thesis. The data sets were built from a number of randomly placed houses in region (X, Y) and a number of ITAPs locations were also placed in region (X, Y).

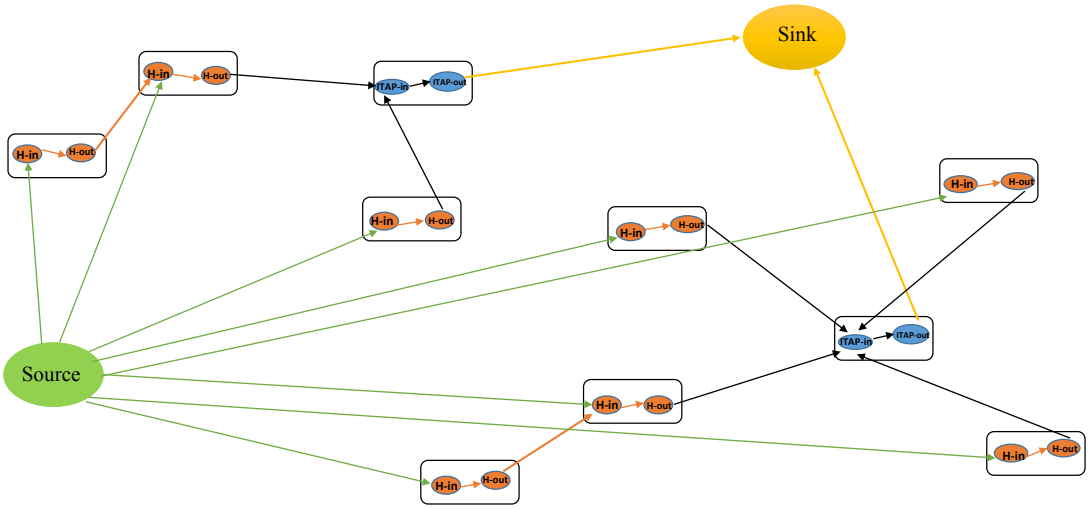

Figure 3.2: Graph G_p for set of houses and ITAPs connected with sink and source.

To create a data set, only the cases where this data sets generated a connected graph in which all houses were connected to at least one ITAP were considered.

Parameter Settings:

The following input parameters were used to generate the data sets:

X, Y	Extent of the simulation region
M	Number of houses
N	Number of ITAP locations
WRC	Wireless range connectivity
WLC	Wireless link capacity
IC	ITAP capacity
h_c	House Capacity
w_h	Demand for each house

The following algorithm 3.1 describes the creation of the data set for the WMN ITAP placement problem. In this algorithm, the houses and ITAPs locations were selected randomly in the (X, Y) region. The wireless nodes were connected to each other if they were within the WRC. The data sets are available if needed at `<https:`

//www.dropbox.com/sh/y55drrkeipvrmpk/AABdft5HuipnZbiLend1rj0Oa?dl=0>

Algorithm 3.1: Data Set Generation

```
Set H = { };
Set I = { };
for i = 1 to M do
    Pick uniformly randomly x ∈ [0, X], y ∈ [0, Y];
    Let h be a house with demand w_h and capacity h_c at location (X, Y);
    H = H ∪ {h};
end
for i = 1 to N do
    Pick uniformly randomly x ∈ [0, X], y ∈ [0, Y];
    Let i be ITAP with capacity IC at location (X, Y) to the set I;
    I = I ∪ {i};
end
Create empty graph with vertex, set H ∪ I;
for each pair of vertices do
    if corresponding wireless nodes are within distance WRC then
        Add edge with weight WLC;
    end
end
```

3.5 Heuristics and Meta-heuristics Applied to the ITAP Placement Problem

In this section, heuristics and meta-heuristics are applied to the problem of placing ITAPs under the constraints of wireless link capacity and wireless range connectivity in the Ideal Link Model. The solution is represented by $s = [y_1, ..., y_N]$.

3.5.1 Greedy Algorithm

Here a simple Greedy algorithm for the WMN ITAP placement problem is considered. This is another heuristic algorithm which is used to build up an optimization solution in a given time to give best flow by adding more ITAPs to the site. The Greedy Algorithm starts with an empty solution and then repeatedly adds an ITAP at the location that provides the greatest improvement that can be obtained by selecting the best ITAP and then picking it as the initial solution. At each iteration the addition of an ITAP is attempted at the best location tracked so far and the amount of satisfied demand in conjunction with ITAPs added in previous iterations is evaluated. To add a new ITAP, the best position so far is noted, as explained in the pseudocode for Algorithm 3.2, below. The Greedy Algorithm decides the location at which to place the first ITAP determined at a particular location by the level of demand from the house and may

face constraints over where to put the ITAPs. C is a constant number of ITAPs.

Algorithm 3.2: Greedy Algorithm (C)

```
1  Set y_i = 0 for 1 ≤ i ≤ N ;
2  while Σ y_i ≤ C do
3      best_i = −1;
4      best = f([y_1, ..., y_N]);
5      for i = 1 to N do
6          Set y_i = y_i + 1;
7          if f([y_1, ..., y_N]) > best then
8              best = f([y_1, ..., y_N]);
9              best_i = i
10         end
11         Set y_i = y_i − 1 ;
12         if best_i != −1 then
13             y_{best_i} = y_{best_i} + 1 ;
14         end
15         else
16             Exit;
17         end
18     end
19 end
```

3.5.2 Hill-Climbing

The Hill Climbing (HC) algorithm is used for finding local optima and explores the search space iteratively by changing a small part of the current solution to get a better solution. In general, HC explores the whole neighbourhood and picks the best. The neighbourhood is defined by all solutions obtained by moving one ITAP to a new location. Stochastic Hill Climbing is used here, which picks one solution randomly in the

neighbourhood. If it is better than the current solution, it moves the ITAP; if not, another random solution is chosen. Stochastic hill climbing avoids the need to calculate objective functions for the entire neighbourhood. The initial solution of the HC can be anything such as a random starting solution or a solution produced by the Greedy Algorithm. Variations of this solution should be tried until a better solution (or at least a non-worse solution) is found, then variations of this new solution should be tried, and so on. HC rarely finds a global optimum, since it is likely to get stuck in a local optimum at a sub-optimal point or plateau which has no superior neighbouring points.

3.5.3 Simulated Annealing

Simulated Annealing (SA) is a meta-heuristic local search method that is more sophisticated than HC. It has been used successfully in solving many combinatorial optimization problems. SA works differently from HC and will occasionally accept worse solutions. SA uses a stochastic optimization method to escape from local minima as the temperature of the system declines. This characteristic of SA helps it to jump out of any local optimums where it might otherwise have become stuck, and it gives a much better chance of finding the global maximum for a given solution

3.6 Definition of Neighbourhood Move Operators

In this chapter, it is proposed that two types of neighbourhood move operator should be applied to the WMN ITAP placement problem. A move operator is the transition process from one solution to another within its neighbourhood. The **neighbourhood move** changes some attributes of the current solution to transform it to a new solution. Local search algorithms with suitable heuristic move operators have been observed to produce acceptable results when the numbers are too large for solutions by exact methods. Thus HC and SA algorithms are tried out as providers of simple frameworks for

experimenting with the proposed move operators. The neighbourhood move operators may help introduce some improvement to the fitness of the current solution. Here "Swap" and "Reallocate" moves were considered. Each of these moves works differently, and the goal is to use both moves together to sample the search space efficiently and effectively. The "Swap" move exchanges the entire allocation of ITAPs at two randomly selected locations, while "Reallocate" moves a single ITAP from one random location to another. Figure 3.3 shows the swap move and reallocates move structures before and after.

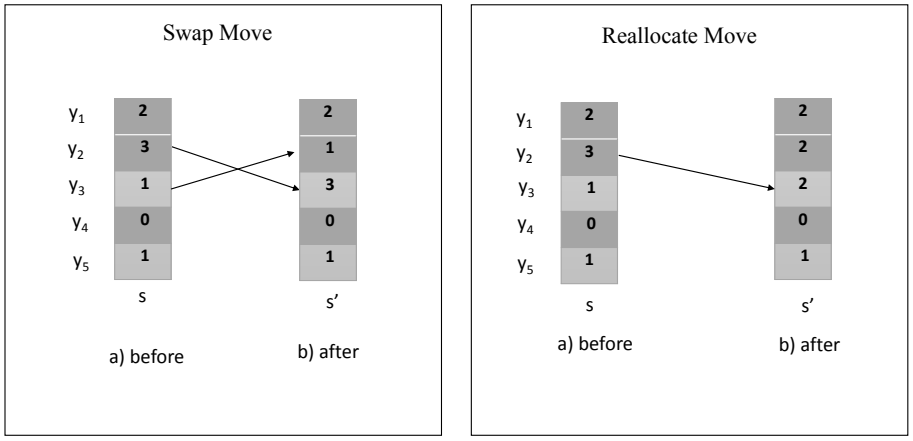

Figure 3.3: Structure of Swap and Reallocate Moves before and after

Algorithm 3.3 shows a pseudocode outline to generate the move, assumes $\mathcal{M}$ as a list of the move operators and represents the s as the initial solution.

3.7 Implementation of HC and SA Algorithms in the Ideal Link Model

The performance of HC and SA Algorithm was evaluated to maximise the throughput by means of the maximum flow algorithm. The "Swap" and "Reallocate" moves were used in order to explore the search space by moving individual ITAPs from one location to another. Note that if the move cannot be made (for example, no ITAPs to reallocate), an entirely new selection is made.

Algorithms 3.4 and 3.5 show the pseudocode of the heuristic and meta-heuristic algorithms.

Algorithm 3.3: Generate $(s, \mathcal{M})$

Function *Generate* $(\boldsymbol{s}, \boldsymbol{\mathcal{M}})$*:* $\boldsymbol{\mathcal{M}}$ *list of move operator function ([swap, reallocate] in this case)*
  Select $m \in \mathcal{M}$ with uniform randomness;
  Return $m(s)$
end
Function *Swap(s):*
  Select i and j randomly from $\{1, ..., N\}$ such that $i \neq j$: ;
  Set $s' = s$;
  if $s'[i] \neq s'[j]$*:* **then**
    $s'[i] = s[j]$;
    $s'[j] = s[i]$;
    Return s'
  end
  else
  Return Generate $(s, \mathcal{M})$;
end
Function *Reallocate(s):*
  Select i and j randomly from $\{1, ..., N\}$ such that $i \neq j$: ;
  Set $s' = s$;
  if $s[i] > 0$*:* **then**
    $s'[i] = s'[i] - 1$;
    $s'[j] = s'[j] + 1$;
    Return s'
  end
  else
  Return Generate $(s, \mathcal{M})$;
end

Algorithm 3.4: HC Algorithm (s, $\mathcal{M}$, max iterations)

```
1 Iteration = 0;
2 while iteration < max iterations do
3 |   s' = generate (s, M);      M list of move operator function;
4 |   Set Δ = f(s') − f(s);
5 |   if Δ > 0 then
6 |   |   Set s = s';
7 |   end
8 end
9 Return s
```

Algorithm 3.5: Simulated Annealing (s, T_0, α, $\mathcal{M}$, max iterations, K)

```
1  T_0 is initial Temperature;
2  α is a parameter to control cooling;
3  K is the Boltzmann constant, used to scale the chance of accepting a worse
   solution;
4  Set best = s;
5  Set f_best = f(s);
6  Set Temperature to initial value T = T_0;
7  Iterations = 0;
8  while number of iterations < max iterations do
9      Set T = α T ;
10     Generate a new solution s' within neighbourhood of s ;
11     s' = generate(s, M) ;
12     Δ = f(s') − f(s) ;
13     if Δ > 0 then
14         Set s = s';
15         if f(s') > f_best then
16             Set best = s';
17             Set f_best = f(s')
18         end
19     else if random ≤ e^{−KΔ/T} then
20         Set s = s';
21     Iterations = iterations + 1
22 end
23 Return best
```

The cost function plays a key role in any optimization problem. It is by calculating this function that one can measure the quality of any solution. Hence, its correct definition is essential for the behaviour of any search algorithm.

3.8 Experimental Results

Algorithms where implemented using Python Language 2.7 with NetworkX library and experiments run on a PC with CPU Intel (R) Core (TM) $i7$ $2.40GHz$ and 12 GB RAM. The Greedy solution can be used as the starting solution. At this point, an HC algorithm should be designed to improve the given starting solution and determine one or more neighbourhood moves. Different wireless ranges (of 25, 30 and 35 meters) were set up in order to produce different levels of connectivity in the underlying graph. Different wireless link capacities (5, 15 and 20 Mbps) were applied to support the traffic flow. The numerical results obtained illustrate the solution solving of the throughput by implementing the HC and SA algorithms. The initial temperature of the SA was set as $T_0 = 2$ and then the temperature was gradually reduced using a cooling schedule, with a temperature reduction of $alpha = 0.85$, and the constant $K = 3$. If alpha is too large then the reduction in the temperature will be very slow and the runtime will be too long. If alpha is too small then the temperature will fall rapidly and the solution is more likely to get stuck in a local minimum. To test the approach effectively, a range of data sets was generated for experimentation, based on a random uniform distribution of nodes. The number of houses, the number of ITAPs, the number of ITAP locations and the grid size were all intentionally varied. When the same number of houses was involved, the houses were placed in the same locations for each data set but with different ITAP locations. Each experiment was run 15 times with different random seed and the average and best result, with the runtime, were reported. The approach of optimization using swap and reallocate moves was applied for the data sets of Table 3.1 in the HC and SA algorithms, which maximise the throughput (traffic flow) by moving ITAPs from one place to another, without changing the number of ITAPs. The aim was

Table 3.1: Benchmark data sets

Data sets	No. of Houses	No. of ITAPs	ITAP Locations	Grid Area
DS1	100	10	10	100×100
DS2	500	50	10	100×100
DS3	500	50	50	500×500
DS4	500	50	70	500×500
DS5	500	50	100	500×500
DS6	500	50	100	1000×1000
DS7	1000	50	100	500×500

Table 3.2: Experiments using data sets

Experiments	Data sets	Wireless Range Connectivity	Wireless Link Capacity	ITAP Capacity	Iterations
E3.1	**DS1**	25	5	10	500
E3.2	**DS2**	25	5	10	1000
E3.3	**DS3**	25	20	20	500
E3.4	**DS4**	25	15	20	500
E3.5	**DS3**	25	15	20	1000
E3.6	**DS5**	25	20	20	500
E3.7	**DS5**	25	15	20	1000
E3.8	**DS5**	30	15	20	500
E3.9	**DS3**	35	15	20	500
E3.10	**DS6**	50	15	20	500
E3.11	**DS2**	25	5	10	500
E3.12	**DS5**	30	15	20	1000
E3.13	**DS3**	35	15	20	3000
E3.14	**DS7**	35	15	20	1000

to demonstrate the effectiveness of these two moves in improving the throughput for a range of data sets with different characteristics (such as wireless range communication, wireless link capacity and ITAP capacity) as shown in Table 3.2.

Table 3.3: Throughput and running time of the Greedy Algorithm

Experiments	Greedy Algorithm	
	Cost	**Run Time/sec**
E3.1	100	3
E3.3	416	16
E3.5	416	18
E3.8	498	25
E3.10	484	22

3.8.1 Experimental Results

The Greedy algorithm was used to generate initial solutions used in HC algorithm, as shown in Table 3.3; it could be seen from the result that this algorithm was faster than other heuristic algorithms.

The Swap and Reallocate moves were tested individually and together. Note that the swap move cannot explore the entire search space, as it is limited to the numbers of ITAPs installed in the first solution. For example, if this has no ITAP location with one ITAP, then there is no mechanism to achieve this allocation. Due to this, we expect swap move to be outperformed by reallocate move. The numerical results in Tables 3.4 and 3.5 include a comparison of the individual move operators, and show that the reallocate move outperforms the swap move but takes longer, as shown in the results **Bold**. Nevertheless, using the swap move reduces the chance of being trapped in a local optimum; hence, the moves are used together to improve the throughput and running time. Comparing the combined moves with the individual move produces the results of the HC algorithm, which shows a slight reduction in the running time as shown in Table 3.4, while the SA algorithm shows a better result in the throughput and running time, as shown in Table 3.5. As expected, the deduction from Tables 3.4 and 3.5 is that the SA algorithm outperforms the HC algorithm in both the individual and the combined moves. It can be observed from the results that the throughput improves

Table 3.4: Maximising Throughput value using the Reallocate and Swap moves individually and together in the HC algorithm.

Experiments	HC					
	Reallocate Move		Swap Move		Reallocate and Swap	
	Cost	Run Time/sec	Cost	Run Time/sec	Cost	Run Time/sec
E3.1	100	7	100	3	100	**6**
E3.3	360	625	345	615	**360**	**623**
E3.5	**364**	1289	354	1265	**362**	**1267**
E3.8	495	1071	494	1040	**495**	**1027**
E3.10	**434**	892	430	866	**432**	**841**

Table 3.5: Maximising Throughput value using the Reallocate and Swap moves individually and together in the SA algorithm.

Experiments	SA					
	Reallocate Move		Swap Move		Reallocate and Swap	
	Cost	Run Time/sec	Cost	Run Time/sec	Cost	Run Time/sec
E3.1	100	69	100	67	100	**66**
E3.3	366	669	365	663	**416**	**665**
E3.5	366	1356	366	1345	**416**	**1332**
E3.8	495	1073	495	1060	**495**	**1045**
E3.10	469	935	461	922	**484**	**889**

when the number of ITAP locations and wireless range connectivity is increased, as seen in Table 3.6 showing Experiments E3.8, E3.9, E3.12, E3.13 and E3.14. The HC and SA algorithms give the same results, showing that HC is sufficient for small and simple data sets, as seen in Table 3.6, of Experiments E3.1, E3.2 and E3.11, but does not work as well as the SA on large sets of data, such as those featuring in Experiments E3.3 - E3.7 and E3.10, which are harder to resolve.

Table 3.6: maximising Throughput in the HC and SA algorithm for combined swap and reallocate moves.

Experiments	HC		SA	
	Cost	Run Time /sec	Cost	Run Time /sec
E3.1	100	6	100	66
E3.2	500	173	500	3623
E3.3	360	623	**416**	665
E3.4	395	762	**425**	797
E3.5	362	1267	**416**	1332
E3.6	431	862	**482**	935
E3.7	440	1712	**482**	1852
E3.8	495	1027	495	1045
E3.9	500	1000	500	1029
E3.10	432	841	**484**	889
E3.11	500	291	500	1811
E3.12	498	2050	498	1391
E3.13	500	2644	500	6168
E3.14	1000	8518	1000	6203

3.9 Network Architecture and ITAP Placement in (Uniform and Grid)

HC and SA algorithms were run on several data sets and optimized the WMN ITAP placement problem. This section suggests that the purpose of the ITAP layout experiments was to justify a random layout which does not hide any unusual features. Two network infrastructures of ITAP deployment were examined, such as uniform and grid layout. A wide range of instances was generated for the experiments, based on a random uniform and grid distribution of ITAPs, with the same number of houses. These houses were placed at the same locations in every instance. Experiments with the heur-

istic algorithm provided a simple framework and results were compared to show the effectiveness of the chosen algorithms in a uniform and a grid ITAP placement. Figure 3.4 shows the sketch of an ITAP placement for the uniform and grid layout of the data set (DS1).

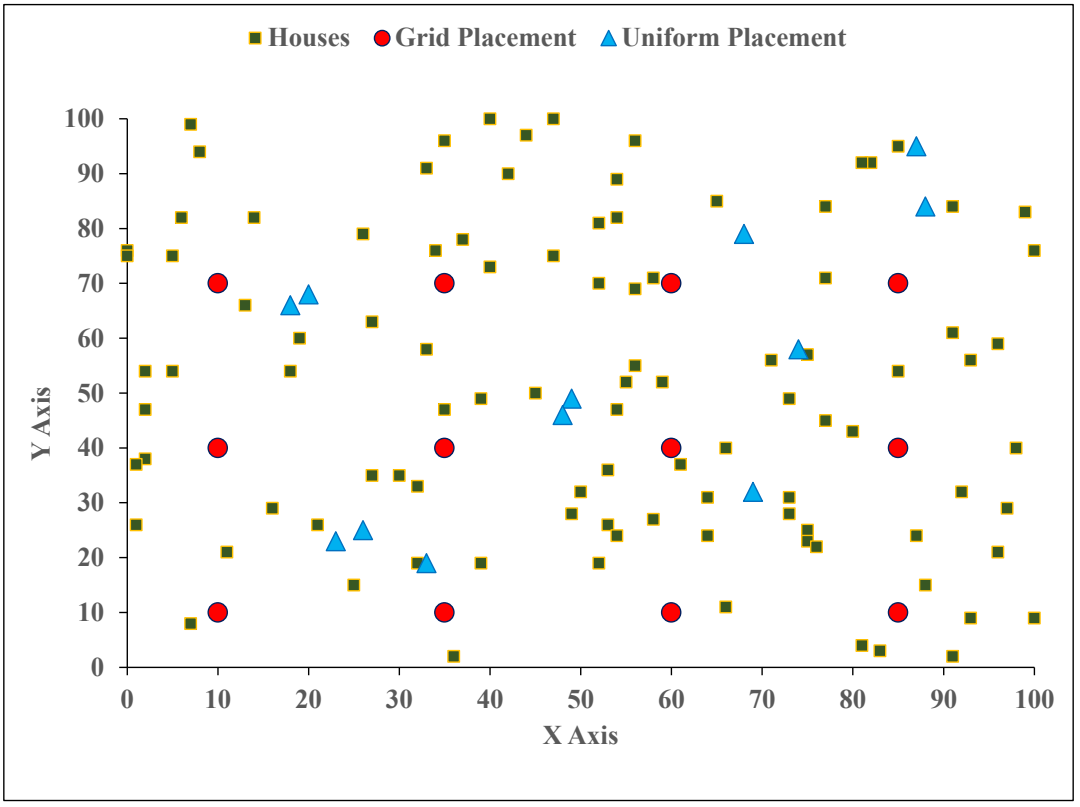

Figure 3.4: Layout of Houses with the Uniform and the Grid ITAP Placements of DS1.

The efficiency of the proposed approach in Table 3.7 was evaluated by checking the results of the Grid and Uniform ITAP placements. From the experimental result it could be seen that the grid and uniform placement showed similar results, varying only a little, as in Experiments E3.1 - E3.5. This indicates that with a small number of ITAP locations, the algorithm works better for grid and uniform layout in maximising the throughput. The instances of small grid area (100×100) give the maximum throughput in both ITAP layouts (for example, Experiments E3.1, E3.2 and E3.11). When the number of ITAP locations increased the uniform placement slightly outperformed the

Table 3.7: Throughput value and running time of Grid and Uniform Placement in the SA algorithm .

Experiments	Grid Placement	Uniform Placement
	Cost	Cost
E3.1	100	100
E3.2	500	500
E3.3	**424**	416
E3.4	**430**	425
E3.5	**424**	416
E3.6	439	**482**
E3.7	439	**482**
E3.8	473	**495**
E3.9	484	**500**
E3.10	460	**484**
E3.11	500	500
E3.12	490	**498**
E3.13	496	**500**
E3.14	1000	1000

grid placement, as shown in Experiments E3.6, E3.7, E3.8, E3.10, and E3.12. Thus, the algorithm in grid placement works well with small data sets, while uniform placement does better with small and large data sets, taken together. Overall, the algorithm for uniform placement gives better results.

3.10 Chapter Conclusion

This chapter has presented the WMN ITAP placement problem formulated as an integer linear programming (ILP). Small and large data sets were generated for exper-

iments with heuristic and meta-heuristic algorithms. Common heuristic and meta-heuristic techniques, such as Greedy, Hill Climbing and Simulated Annealing were reviewed in this chapter. The Greedy Algorithm was implemented to obtain the initial solution; its experimental results showed that it is faster than other heuristic and meta-heuristic algorithms. In practice, the problem cannot be solved by an exact IP solver such as CPLEX, because of the limitations in solving large problems. CPLEX can handle only small networks and is clearly too slow to use on large ones. Hence, approaches were developed to optimize the mesh network problem. An approach and a heuristic move operator have, however, been developed for solving the WMN ITAP placement problem. This heuristic move operator involves the moves of "Swap" and "Reallocate". The approach has been shown to be highly successful for optimizing ITAP placements in WMNs. It can be verified from the experimental results that the reallocate move always performed better than the swap move, even though it was slower. The effectiveness of the proposed algorithms for the ITAP placement of Uniform and Grid layouts was assessed, demonstrating by the experimental results that the algorithms used with the uniform and grid ITAP placements in different data sets were efficient. According to the results, the SA in use produces better results than the HC, but takes slightly longer to run. Therefore, in some experiments the implementation of SA algorithm achieved the best possible solution, giving full coverage of the bandwidth, whereas in other cases good solutions were hard to find. Experimental evaluation showed the efficiency of combining the two move operators; this provided a better solution for the placement of ITAPs in WMNs. Having shown the suitability of meta-heuristic techniques, we will consider other aspects of modelling the problem.

Chapter 4

Optimization of ITAP Placement to Minimise the Number of ITAPs

Chapter 3 considered the problem of maximising the demand that could be supported by a specific number of ITAPs. This chapter demonstrates the effectiveness of novel move operators within the framework of heuristic and meta-heuristic single objective optimization techniques in dealing with the ITAP placement problem. The aim is to minimise the number of ITAPs while satisfying the users' bandwidth requirements as a hard constraint. The proposed approach shows that rapid improvements to the cost of a WMN can be made simultaneously, by using an effective combination of move operators. This chapter is structured as follows: Section 4.1 defines the Integer Linear Programming Formulation of the ITAP placement problem, while section 4.2 defines the neighbourhood move operators proposed for solving the WMN bandwidth optimization problem. Section 4.3 provides a framework for implementing Hill Climbing and Simulated Annealing in optimizing ITAP placement. Section 4.4, details the findings when implementing the move operators. Section 4.5 elaborates on some further analysis of the impacts of parameters on network performance. Finally, section 4.6 concludes this chapter.

4.1 The ITAP Placement Problem

In Chapter 3, the number of ITAPs to be deployed was a hard constraint, whereas in this chapter the throughput demanded by houses should be satisfied as a hard constraint and attempts should be made to minimise the number of ITAPs. The ideal link model used in Chapter 3 is modified to reflect this in the equations and inequalities listed below.

The general objective is to minimise the number of ITAPs while satisfying users' demand for bandwidth. The inequality, equation 4.2 formulates the constraint that each house has w_h amount of flow sent. The rest of the equations 4.3 - 4.9 are identical to Formulation 3.1 described in Chapter 3.

Formulation 4.1:

Minimise $\sum_{1 \leq i \leq N} y_i$

Subject to:

$$\sum_{e=(v,h')} x_{e,h} = \sum_{e=(h',v)} x_{e,h} \qquad \forall h, h' \in H, h' \neq h \tag{4.1}$$

$$\sum_{e=(v,h)} x_{e,h} \geq w_h \qquad \forall h \in H \tag{4.2}$$

$$\sum_{e=(v,h)} x_{e,h} = 0 \qquad \forall h \in H \tag{4.3}$$

$$\sum_{h} x_{e,h} \leq Cap_e \qquad \forall e \in E(G) \tag{4.4}$$

$$\sum_{h',e=(v,h)} x_{e,h'} \leq Cap_h \qquad \forall h \in H \tag{4.5}$$

$$\sum_{h',e=(v,i)} x_{e,h'} \leq Cap_i y_i \qquad \forall i \in I \tag{4.6}$$

$$\sum_{e=(v,i)} x_{e,h} \leq w_h y_i \qquad \forall i \in I, h \in H \tag{4.7}$$

$$x_{e,h} \geq 0 \qquad \forall e \in E(G), h \in H \tag{4.8}$$

$$y_i \in \{0, 1, 2, ...\} \qquad \forall i \in I \tag{4.9}$$

4.2 The Modified Neighbourhood Move Operators in ITAP placement

Following the same principle as Chapter 3, the neighbourhood is considered first. The methodology of a neighbourhood move is to pick a nearby solution and evaluate it and then decide whether to accept or reject that solution. The same algorithm should continue to be used for each iteration. Here two additional neighbourhood move operators are proposed, which allow exploration of the search space. The operators are a Delete move, which removes an ITAP from a random location, and an Add move which increases the number of ITAPs at a random location. The structure of the delete and add moves before and after are shown in Figure 4.1. These moves are in addition to those introduced in Chapter 3, namely, the swap and reallocate moves. The goal is to use combinations of moves to sample the search space more efficiently and effectively. The cost function is to minimise the number of ITAPs. As in Chapter 3, a solution is represented by a vector $[y_1, ..., y_N]$ where y_i is the number of ITAPs to be installed at location i. A move modifies a solution s to a new solution s'.

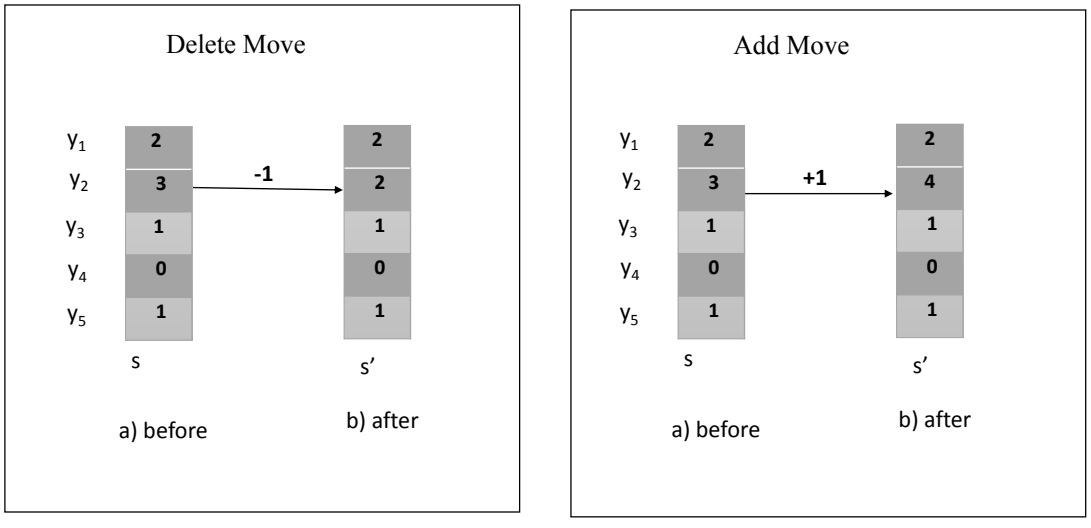

Figure 4.1: Structure of Delete and Add moves before and after

4.3 HC and SA in Optimizing ITAP Placement

HC and SA algorithms are again applied with a combination of move operators to solve the WMN bandwidth optimization problem. The performance of HC and SA algorithms in maximising throughput and reducing the number of ITAPs is evaluated.

Algorithm 4.1 shows the pseudocode outlines of the four moves; i.e. Swap, Reallocate, Delete and Add.

Algorithm 4.1: Generate $(s, \mathcal{M})$

Function *Generate* $(s, \mathcal{M})$*:* $\mathcal{M}$ *list of move operator functions*
    Select $m \in \mathcal{M}$ with uniform randomness;
    Return $m(s)$
end
Function *Swap (s):*
    Select i and j randomly from $\{1, ..., N\}$ such that $i \neq j$: ;
    Set $s' = s$;
    if $s'[i] \neq s'[j]$*:* **then**
        $s'[i] = s[j]$;
        $s'[j] = s[i]$;
        Return s'
    end
    else
    Return Generate $(s, \mathcal{M})$;
end
Function *Reallocate (s):*
    Select i and j randomly from $\{1, ..., N\}$ such that $i \neq j$: ;
    Set $s' = s$;
    if $s[i] > 0$*:* **then**
        $s'[i] = s'[i] - 1$;
        $s'[j] = s'[j] + 1$;
        Return s'
    end
    else
    Return Generate $(s, \mathcal{M})$;
end

```
1  Function Delete (s):
2      Select i randomly from {1, ..., N} : ;
3      Set s' = s;
4      if s[i] > 0: then
5          s'[i] = s[i] − 1;
6          Return s'
7      end
8      else
9      Return Generate (s, M);
10 end
11 Function Add (s):
12     Select i randomly from {1, ..., N};
13     Set s' = s;
14     s'[i] = s[i] + 1;
15     Return s'
16 end
```

The pseudocode outlines of the heuristic and meta-heuristic algorithms described in Algorithm 4.2 and 4.3. The only change from Chapter 3 is the direction of optimization regarding minimisation instead of maximisation. We also use the cost function $f([y_1, ..., y_N])$ = $\sum_{1 \leq i \leq N} y_i$.

4.4 Experimental Results

Experiments were carried out with the HC and SA algorithms with the proposed move operators: Swap, Reallocate, Delete and Add. These moves are all designed to work in different ways on the same aspects of the solution to the number of ITAPs at the loca-

Algorithm 4.2: HC Algorithm (s, $\mathcal{M}$, max iterations)

```
1 Iterations = 0;
2 while iterations < max iterations do
3     s' = generate(s, M) #Generate a new solution s' within the neighbourhood of
        s ;
4     Set Δ = f(s') − f(s)
5     if Δ ≤ 0 then
6         Set s = s';
7     end
8 end
9 Return s
```

tions, and the goal is to use combinations of the moves to sample the search space most efficiently and effectively. To minimise the number of ITAPs and maximise throughput, the NetworkX was applied with a minimum flow algorithm. More data sets were generated, following the same processes described in Chapter 3, to test the approach; some of the data sets in Chapter 3 were tested in the experiments, as shown in Tables 4.1 and 4.2, on the basis of a random uniform distribution of nodes. The number of houses, the number of ITAP locations and the grid size were all varied. For example, all the houses were placed at the same locations for each data set. Each experiment ran 15 times with different random seeds and the average and best result, together with the runtime, were reported. The approach was applied using all four move operators. The main goal here was to minimise the number of ITAPs. The effectiveness of the four moves in improving the throughput was demonstrated by using the "Swap" and "Reallocate" move and the number of ITAPs was minimised by using the "Add" and "Delete" moves on a range of data sets with different characteristics (such as ITAP capacity, the number of ITAP locations, wireless link capacity and wireless range communication).

Different starting solutions were investigated by placing an initial number of ITAPs

Algorithm 4.3: SA Algorithm (s, T_0, α, $\mathcal{M}$, max iterations, K)

T_0 is initial Temperature;
α is a parameter to control cooling;
K is the Boltzmann constant, used to scale the chance of accepting a worse solution;
Set $best = s$;
Set $f_{best} = f(s)$;
Set Temperature to initial value $T = T_0$;
Iterations = 0;
while *iterations* < *max iterations* **do**
    Set $T = \alpha T$;
    s' = generate $(s, \mathcal{M})$ #Generate a new solution s' within the neighbourhood of s;
    $\Delta = f(s') - f(s)$
    if $\Delta \leq 0$ **then**
        Set $s = s'$;
        if $f(s') \leq f_{best}$ **then**
            Set $best = s'$;
            Set $f_{best} = f(s')$
        end
    else if $random \leq e^{-K\Delta/T}$ **then**
        Set $s = s'$;
    Iterations = iterations + 1
end
Return $best$

randomly (i.e. 40, 50 and 60). The experimental results in Table 4.3 show that using combinations of moves is able to reduce the number of ITAPs that are required to satisfy the demands. However, the cost function that was used for these experiments

Table 4.1: Benchmark data sets

Data sets	No. of Houses	No. of ITAPs	ITAP Locations	Grid Area
DS2	500	50	10	100×100
DS3	500	50	50	500×500
DS5	500	50	100	500×500
DS7	1000	50	100	500×500

Table 4.2: Experiments on data sets

Experiments	Data sets	Wireless Range	Wireless Link	ITAP Capacity	Iterations
E4.1	**DS2**	25	5	10	500
E4.2	**DS5**	30	15	20	1000
E4.3	**DS3**	35	15	20	3000
E4.4	**DS7**	35	15	20	1000

shows that there is a trade-off between minimised ITAPs and unsatisfied houses. An improvement in reducing the number of ITAPs can be observed from the results, when the number of ITAP locations and the wireless range connectivity are increased, as seen in E4.2 and E4.3 for both algorithms. The greatest reduction of ITAPs was effected by using the SA Algorithm.

The experimental results confirm the evaluation of the combination of move operators with the SA algorithm as good and efficient in satisfying the demand from the houses with fewer ITAPs, as shown in Table 4.3. The cost function of E4.2 and E4.3 showed the efficiency with a smaller number of ITAPs. Thus, the cost function in SA surpasses HC and yields excellent solutions by effectively combining different move operators, as seen in Table 4.3 in **bold**. Figure 4.2 shows the result of E4.2 starting with different numbers of ITAPs (40, 50 and 60). This shows that the HC struggles to move away from the initial solution and thus does not change the number of ITAPs significantly, but the SA made a reduction in the number of ITAPs and finds a consistent solution.

Table 4.3: Cost function of the data sets using all move operators with a different number of ITAPs in the HC and SA Algorithms.

Experiments	HC				SA			
	Cost Min (Avg)	RT/sec	Satisfied Houses	#ITAP End (start)	Cost Min (Avg)	RT/sec	Satisfied Houses	#ITAP End (start)
E4.1	590(596)	237	500	59(60)	500(500)	1697	500	50(60)
E4.2	735(755)	1555	495	58(60)	425(441)	2131	495	**27(60)**
E4.3	580(594)	1050	500	58(60)	250(250)	5962	500	**25(60)**
E4.4	600(600)	5320	1000	60(60)	500(500)	8569	1000	50(60)
E4.1	500(500)	175	500	50(50)	500(500)	1759	500	50(50)
E4.2	635(667)	1545	495	48(50)	415(425)	2152	495	**26(50)**
E4.3	470(496)	741	500	47(50)	250(250)	6099	500	**25(50)**
E4.4	500(500)	2152	1000	50(50)	500(500)	9006	1000	50(50)
E4.1	3200(3380)	153	410	41(40)	500(500)	1755	500	50(40)
E4.2	545(733)	1421	495	39(40)	415(423)	2161	495	**26(40)**
E4.3	370(396)	2747	500	37(40)	250(250)	6167	500	**25(40)**
E4.4	5380(6234)	3723	840	42(40)	500(500)	9080	1000	50(40)

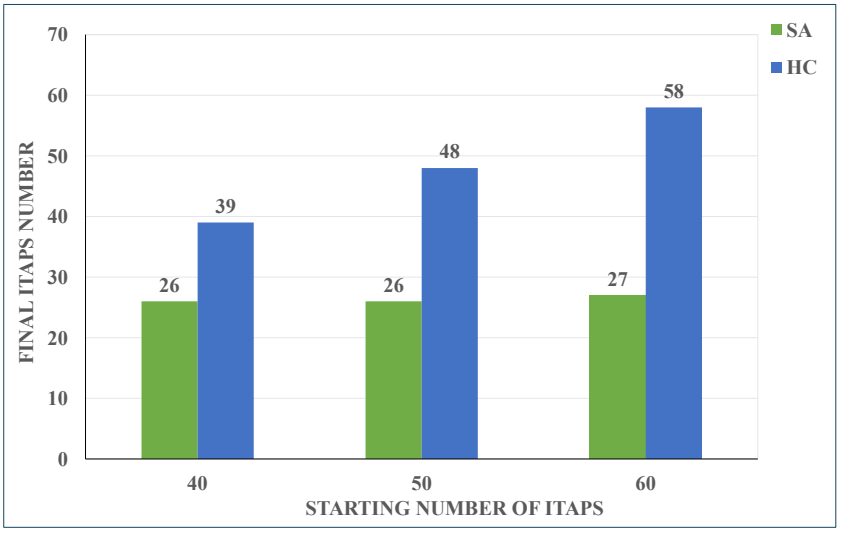

Figure 4.2: Result of E4.2 starting with different numbers of ITAPs (40, 50 and 60).

4.5 Effect of parameters on network performance

Different parameters were used in the network optimization problem and these parameters have impacts on the wireless networking performance. The performance could include the quality of the solution returned and the time taken by the algorithm. Many factors, depending on areas within the network itself, affect the wireless networking performance, such as the technology of the devices used, the local environment and the fundamental physics behind wireless transmission. Some of these cannot be avoided and action can only be taken to minimise the negative effects that these factors will have on the network performance, but others can be resolved completely either through equipment upgrading or good network planning. Many parameters and constraints of the model need to be taken account of, since they impact on the wireless network performance in providing optimal solutions such as:

1. **Network Size:** The impact of network size on the placement algorithms should be considered. There is a random distribution of nodes (houses) in an $N \times N$ area. The evaluation results show that an increase in the network size will increase the number of ITAPs required. As might be expected, an increase in the grid size leads to a greater number of ITAPs being required to cover the region, which is spread over a large grid area (see E4.2, E4.3 and E4.4).

2. **Neighbourhood Topology:** The placement algorithms and number of houses, which have a big impact on the network performance, should be evaluated. The experimental evaluation indicates that many houses are connected with each other through a high communication range, leading to a single connected component [1] of the neighbourhood. For instance, comparing E4.4 with other experiments shown in Table 4.3 shows that a smaller number of ITAPs is required,

[1] Sometimes a graph may not be fully connected or it may have groups of vertices that are closely connected. Hence, a connected component is a maximal connected sub graph of G. Each vertex belongs to one connected component only, as does each edge

since the houses will be close to each other and they can be connected as one component, which will then be of high capacity. Conversely, a smaller number of houses with a small communication range will lead to multi connected components in this neighbourhood, in which case a high number of ITAPs is required. Additionally, the more nodes installed, the higher the network coverage in the neighbourhood, as indicated in E4.4.

3. **ITAP Capacity:** Increasing ITAP capacity will proportionally decrease the required number of ITAPs. The experimental results demonstrate the high flow and number of ITAPs required when increasing the capacity of these ITAPs, as shown in E4.2.and E4.3, whereas reducing ITAP capacity increases the required number of ITAPs, as shown in E4.1. The ITAP can serve nodes as long as the sum of their demands does not exceed the capacity of the ITAP.

4. **Wireless Range Communication:** This is the distance of coverage between two nodes in the network, measured in meters. In the experiment a fixing communication radius between 25 and 35 meters was applied. Consequently the increase in communication range would create much overlap in the wireless coverage of the houses, as noted above. Therefore with a high wireless range fewer ITAPs are required to satisfy the houses' demands, as shown in E4.3 where all the houses are satisfied with a smaller number of ITAPs. If the communication range is very small, most houses are disconnected from one another and the number of ITAPs required is nearly the number of houses, as in E4.1. A high number of ITAPs is then required to satisfy the demand from the houses. Thus the number of required ITAPs decreases with increased communication range. This can be illustrated from the experimental results.

5. **Wireless Link Capacity:** The wireless link capacity can make a big impact on the network performance. When the wireless bandwidth is equal to a single house's demand, the number of ITAPs required is significantly large, as in E4.1, with small wireless link capacity of 5 Mbps a large number of ITAPs are needed

to satisfy the demand from all the houses. With an additional increase in the wireless capacity, the link is no longer a bottleneck and the overall bandwidth can increase; then the number of required ITAPs remains the same as shown in E4.2 - E4.4. Hence, the wireless link capacity has a big impact on the number of ITAPs required.

4.6 Conclusion

This chapter has presented an approach for solving the WMN bandwidth optimization problem, using heuristic move operators. It has been demonstrated that the approach is highly successful for optimizing ITAP placements in WMNs. Clearly, the application of the correct neighbourhood move operator is essential to the success of the search algorithm. The experimental evaluation shows the efficiency of a combination of all four move operators because it provides a better solution for the placement of ITAPs in WMNs. The results show that the SA produces better results than the HC in satisfying the houses' demand with fewer ITAPs, but takes longer to run. The parameters used in the model showed a big impact on the network performance in optimizing ITAP placement as regards the maximising of throughput and minimising of the number of ITAPs.

Chapter 5

Optimizing Bandwidth Allocation in WMN

Chapter 3 and Chapter 4 considered the problem of optimizing the number of ITAPs and throughput respectively (while keeping the other fixed). Such problems are known as single objective optimisation problems, where a single objective function is to be minimised or maximised while satisfying a number of hard constraints as in [57]. Many practical problems, however, need to optimize several objectives simultaneously in order to achieve a desired result. For example, here there is a direct trade-off between the number of ITAPs deployed and the throughput that can be achieved. This type of problem is known as a multi-objective optimization problem (MOO). In this chapter we address the WMN infrastructure placement optimization problem as a multi-objective approach with three conflicting objectives:

- Minimising the number of Internet Transit Access Points (ITAP)
- Maximising the fairness of bandwidth allocation to users
- Maximising the total throughput delivered in the network

We apply the weighted-sum method and implement a metaheuristic algorithm, using also the novelty of an efficient combination of move operators to solve the problem. This algorithm produces a set of effective optimization solutions under the ideal link network model, as described in Chapter 3.

In the present chapter we briefly describe in section 5.1 the objective functions, throughput and fairness of the bandwidth and the trade-off between the objectives. Section 5.2 briefly highlights the integer linear program formulation of our problem. We then proceed in section 5.3 to review important techniques that are usually applied in solving optimization problems of WMN infrastructure placement, using the weighted-sum method. A Simulated Annealing algorithm for ITAP placement and bandwidth allocation, and a combination of moves that may be applied in the search for a good solution are introduced in section 5.4. Section 5.5 describes the normalization of the objective functions. Section 5.6 reports the experimental results of the best moves investigated for a range of benchmark problems. Finally, section 5.7 concludes the chapter.

5.1 Objective Functions

The objective of minimising the number of ITAPs placed is again used to model the additional infrastructure cost required to serve additional houses. Here we define, as described in Chapter 3, the measures of throughput and fairness in wireless mesh networks.

5.1.1 Throughput

Throughput refers to the amount of data that can be transferred from one location to another in a given time period. Real traffic typically varies over many or all time scales, and is characterised by its bursts (from inconsistent traffic levels). Although traffic typically occurs in bursts, for the infrastructure placement problem considered here, we consider data rates averaged over a long time period, of the order of seconds, in effect averaging and smoothing out the bursts over a given time interval.

5.1.2 Fairness

In WMNs fairness is generally attributed to bandwidth allocation. Fairness is used to determine whether users are receiving a fair share of bandwidth allocation. Different models of fairness can be applied to WMN, as discussed in Chapter 2. Achieving fairness in WMNs is equivalent to solving a maximisation problem, subject to the transmission constraints. Maintaining fairness in WMNs is fairly important but has been given less attention than other attributes, such as capacity maximisation and connectivity. Hence, we consider fairness as one of the crucial objective functions. In the interests of fairness, it is preferable to allocate the bandwidth as evenly as possible. The significance of a bandwidth unfairly allocated between different individual users is that it may lead to bandwidth starvation. Bandwidth starvation occurs when a house is not being served, that is when no bandwidth is available to one or more houses. Thus, we look into bandwidth allocation in order to achieve a good trade-off between fairness and throughput. Providing equal amounts of throughput for nodes far and near to ITAPs is fairer than not doing so. There are also considerable issues surrounding fairness in WMN, such as the fact that nodes far from the ITAPs have a more significant effect on congestion in the network.

Our model of fairness is to allocate a proportion of the demand from each house. To calculate fairness, one needs to work out the minimum flow of traffic attributable to each house in relation to all the traffic that flows, as shown in the equation below.

$$fairness = \min_{1 \leq h \leq M} (\sum\nolimits_{e=h,v} x_{e,h}) \tag{5.1}$$

5.1.3 Trade-off between objectives

We consider three interdependent objectives in our optimization problem; improving one of these may have a negative effect on the other objectives. We can add more infrastructure to reduce unserved demand and to increase fairness; for example, if we increase the number of ITAPs we will reduce unserved demand but if we increase the

number of houses served, this may reduce fairness. This highlights the fundamental problem of forwarding. Regarding the distance, far nodes have a dramatic effect on capacity, as seen in Figure 5.1, which shows that the far nodes require forwarding which reduces the capacity available for traffic on all links on the path to the ITAP.

In order to support major possible applications such as internet broadband access, it is important to allocate the limited bandwidth fairly to all users. We look at the trade-off between the nodes that are far from and near to the ITAPs; if, for instance, we allocate high bandwidth to one house close to the ITAP (which now has access to all the traffic it needs), it would give high throughput and no contention on the link, but would be very unfair to the other houses. Equally, allocating everyone a fair proportion of traffic would have a significant effect because 'this action' has to forward and increase capacity, and might need more ITAPs and reduce the overall throughput that is delivered. The bandwidth becomes a bottleneck and yet has to transfer the heaviest traffic in the network. The presence of these bottlenecks reduces the available capacity for each house. Hence, the model of fairness that we select to be fair to everyone, giving them all the same proportion of their demand, will regulate the throughput of the entire network. In some cases, to maximise fairness we have no need to maximise throughput, but could provide the same throughput to all nodes, whether far or near, which would result in a good trade-off between throughput and fairness. In Chapter 3 we investigated where to place the ITAPs in order to maximise throughput, so it is important to find the optimal ITAP placement that will maximise the throughput. Figure 5.1 shows the connection of far and near nodes to the ITAPs: the blue nodes represent the ITAPs, with each ITAP consisting of in and out nodes as virtual nodes. An orange node represents a house which also has an in and an out node as virtual nodes. The oval outline represents the wireless range of connectivity, which means that all the nodes in the same oval are within communication range of each other.

We have three case scenarios to discuss:

- Increasing the number of houses

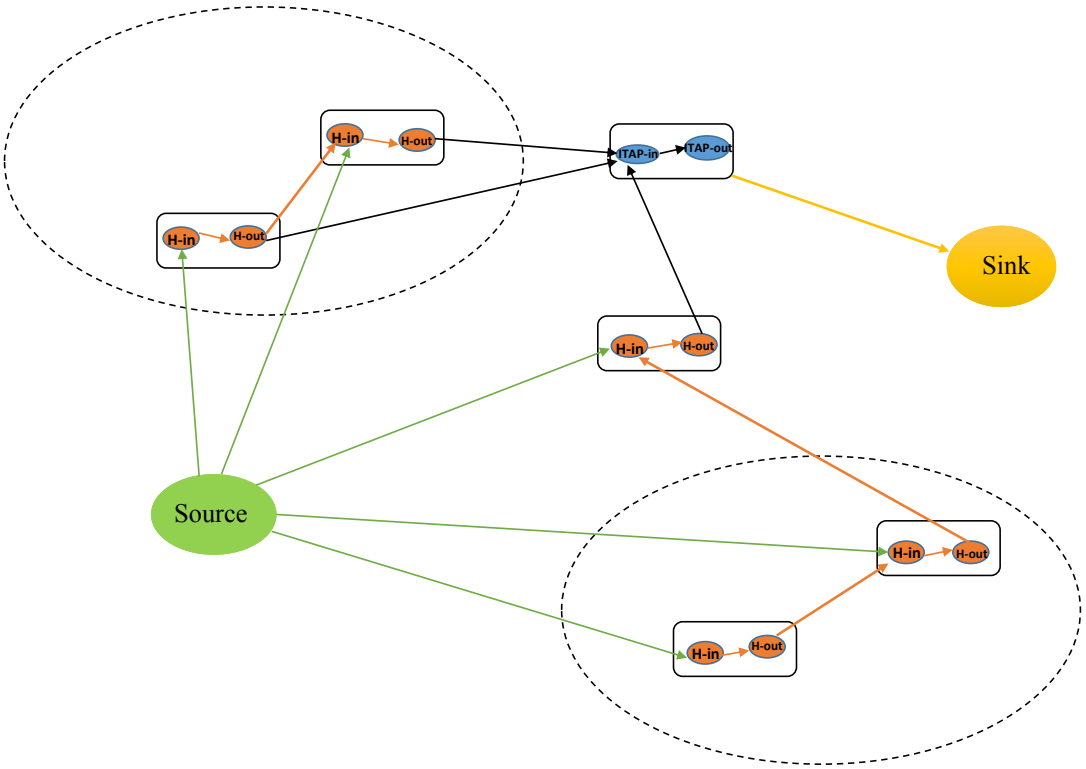

Figure 5.1: Nodes far from and near to the ITAPs in the search space

- Increasing the number of houses and the number of ITAPs
- Increasing the number of ITAPs and determining their placement

The first one, which increases the number of houses, is the base case as seen in Figure 5.1 which shows the connection of far and near nodes in relation to the ITAP. This scenario demonstrates the problem of scaling the number of houses in terms of increasing the demand that cannot be met by ITAPs. For example, H5 has the sum of traffic from H3, H4 and H5, as shown in Figure 5.2. In the second scenario, we increase the number of houses and number of ITAPs and this is the scale base case that increases the number of subscribers, as shown in Figure 5.3. There is a trade-off between nodes when the traffic of far and near nodes is forwarded. The third scenario is to have an increasing number of ITAPs and determine their placement; this demonstrates the trade-off that results from adding ITAPs and determining their placement so as to serve more houses.

Figure 5.2 shows the parameters of capacity for the edges, houses and ITAPs in these scenarios; these capacities relate to the equations in Formulation 5.2.

Adding houses further away from the ITAPs has a dramatic reductive effect on capacity. Figure 5.3 shows the trade-off entailed by adding ITAPs and serving more houses.

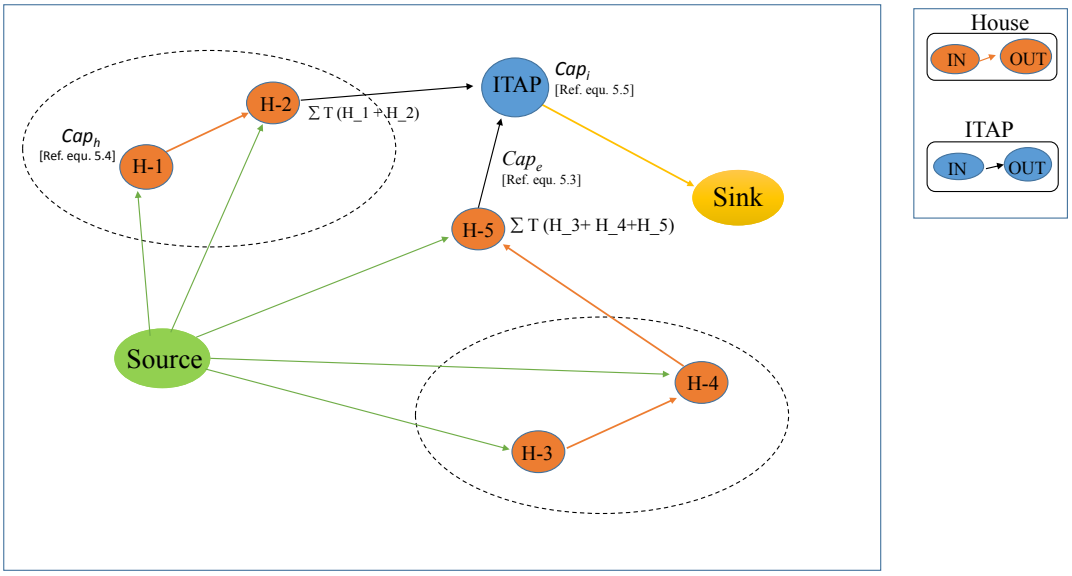

Figure 5.2: Far and near nodes in relation to ITAPs in the search space

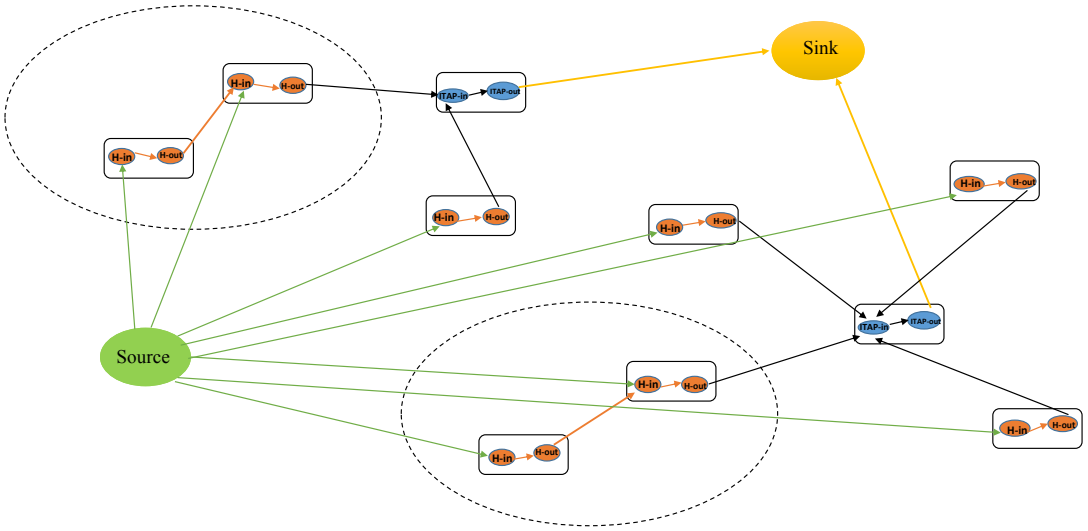

Figure 5.3: Trade-off between ITAPs and houses to ease the traffic

5.2 Integer Linear Program Formulation

As mentioned before in Chapter 3, we use the ideal link model proposed in [2], and aim to minimise the number of ITAPs required, while maximising users' bandwidth requirements with the fairest possible allocation. In Chapter 4, algorithms are applied to place ITAPs at different locations, with a minimum flow algorithm to model the flow of traffic through the network. Before applying a minimum flow algorithm, however, it is necessary to transform the WMN into an equivalent single source, single sink model. Once this has been done, the minimum flow algorithm can be used to compute the maximum flow capacity of the edges in the network flow, and hence to determine whether or not a given WMN configuration is able to support the user demand in practice. Fairness in the network can be manipulated by adding a series of variables to limit the maximum bandwidth that we allow each user to receive. To provide a normalised set of decision variables, we model this bandwidth allocation as a proportion of the traffic demand from each house. Making the bandwidth b_h means that the bandwidth allocated to each house will be at most $b_h W_h$. In this section, we formally describe the model and define the variables and constraints as shown in Formulation 5.2. Constraints 5.2 - 5.9 are identical to those described in Chapter 3. Constraint 5.10 specifies that the flow from a house must be equal to or less than demand scaled by the allocation for the house. Constraint 5.11 indicates the proportion of demand that each house is allocated, specifically between [0, 1].

Formulation 5.2:

Minimise $W_l\, f_l(x) + W_D\, f_D(x) + W_U\, f_U(x)$

$\sum_{e=(v,h')} x_{e,h} = \sum_{e=(h',v)} x_{e,h} \qquad \forall h, h' \in H, h' \neq h$

Subject to:

$$\sum\nolimits_{e=(v,h')} x_{e,h} = \sum\nolimits_{e=(h',v)} x_{e,h} \qquad \forall h, h' \in H, h' \neq h \tag{5.2}$$

$$\sum\nolimits_{e=(v,h)} x_{e,h} = 0 \qquad \forall h \in H \tag{5.3}$$

$$\sum\nolimits_{h} x_{e,h} \leq Cap_e \qquad \forall e \in E(G) \tag{5.4}$$

$$\sum\nolimits_{h',e=(v,h)} x_{e,h'} \leq Cap_h \qquad \forall h \in H \tag{5.5}$$

$$\sum\nolimits_{h',e=(v,i)} x_{e,h'} \leq Cap_i y_i \qquad \forall i \in I \tag{5.6}$$

$$\sum\nolimits_{e=(v,i)} x_{e,h} \leq w_h y_i \qquad \forall i \in I, h \in H \tag{5.7}$$

$$x_{e,h} \geq 0 \qquad \forall e \in E(G), h \in H \tag{5.8}$$

$$y_i \in \{0, 1, 2, ...\} \qquad \forall i \in I \tag{5.9}$$

$$\sum\nolimits_{e=(h,v)} x_{e,h} \leq b_h w_h \qquad \forall h \in H \tag{5.10}$$

$$b_h \in [0, 1] \qquad \forall h \in H \tag{5.11}$$

5.3 Weighted Sum Approach

We provided a brief description of the weighted-sum approach in Chapter 2, a common approach used in MOO on account of its simplicity, which combines all objectives into a single scalar value to be optimized as a single objective. The objectives have different units, with different numerical ranges, making it difficult to choose the appropriate weights to control the contribution of each objective in relation to the weighted total. The objective function of Formulation 5.2 represents a sum of the weighted normalized objectives. In our experiment we change the value of weights in order to investigate the Pareto fronts and the trade-off between the three objectives. The weighted sum approach means that we have to set arbitrary weights which are used to balance the objectives in advance of optimization. We formulate the WMN infrastructure placement optimization problem of the ITAP placement and bandwidth allocation problem for the ideal link model as an integer program, as shown in Formulation 5.2 , to minimise the composite objective function in (5.12)

$$f(x) = W_I \, f_I(x) + W_D \, f_D(x) + W_U \, f_U(x) \tag{5.12}$$

We express the set of weights as (W_I, W_D W_U), denoting the relative importance of optimizing separately the number of ITAPs, unserved demand and unfairness. The weight of each objective function is in the range [0, 1] and the weights are chosen such that.

$$W_I + W_D + W_U = 1 \tag{5.13}$$

5.4 ITAP Placement and Bandwidth Allocation using Simulated Annealing

In practice we could not solve Formulation 5.2 by an exact IP solver such as Cplex, since it is too slow to use on large examples; hence, we again utilize the Simulated Annealing (SA) algorithm. We apply SA straightforwardly to this problem, proposing novelty in the moves to ensure a balance between optimality and diversity in the search.

Decision variables are encapsulated in a vector $x = [y_1, ..., y_N, b_1, .., b_M]$, where N denotes the number of ITAP locations, M denotes the number of houses and b_i denotes the amount of bandwidth allocated to house i. $U(s)$, represents a uniformly random selection from the set s.

In Algorithm 5.1, we provide pseudo code outline of the SA method that assumes "$\mathcal{M}$" to be a list of move operators. This element of novelty, bringing diversity to the search space by different types of neighbourhood move operators to maximise the network flow and minimise the number of ITAPs in these locations, is described in Chapters 3 and 4.

Initially, in Chapter 4 we used an SA algorithm to provide a simple framework for experimenting with our proposed move operators: Swap, Reallocate, Delete, and Add. Each of these moves is designed to work on different ways on the same aspects of the candidate WMN configuration, and our goal was to use combinations of our moves to sample the search space efficiently and effectively.

Algorithm 5.1: Simulated Annealing ($s, T_0, \alpha, \mathcal{M}$, max iterations, K)

#T_0 is initial Temperature;
#α is a parameter to control cooling;
#K is the Boltzmann constant, used to scale the chance of accepting a worse solution;
Generate an initial solution ;
$s = [y_1, ..., y_N, b_1..., b_m]$;
Set $best = s$;
Set $f_{best} = f(s)$;
Set Temperature to initial value $T = T_0$;
Iterations = 0;
while *number of iterations* < *max iterations* **do**
    Set $T = \alpha\, T$;
    #Generate a new solution s' within the neighbourhood of s ;
    s' = generate(s, $\mathcal{M}$);
    $\Delta = f(s') - f(s)$
    if $f(s') \leq f_{best}$ **then**
        Set $s = s'$;
        Set $best = s$;
        Set $f_{best} = f(s)$
    else if $U([0, 1]) \leq e^{-K\Delta/T}$ **then**
        Set $s = s'$;
    Iterations = iterations + 1
end
Return $best$

The “Swap placement” move exchanges the entire allocation of ITAPs at two randomly selected locations, while “Reallocate placement” moves a single ITAP from one random location to another. “Delete placement” removes an ITAP from a random

location, and "Add placement" increases the number of ITAPs at a random location. In this chapter, similar moves are defined here for the bandwidth allocated to each house i.e. "Reallocate Allocation, Swap Allocation, Delete Allocation, and Add Allocation". Since the number of houses is typically large, and it is difficult to move from one solution to another around the search space, a good set of moves that lets us move from one area to another fairly quickly was applied. We further define the moves that make larger changes possible. The adding aggressive move "Add delta allocation" increases the bandwidth allocation of all houses by a randomly chosen value $\Delta \in [0, 1]$, while the deleting aggressive move "Delete delta allocation" removes bandwidth allocation from all houses simultaneously.

The general mechanism of the move operator is to select a random move from the move operator uniformly randomly. If the selected move was "Delete Placement", removing an ITAP from a random location, and if the selection was invalid (such that no ITAPs were removed), then we should go back to the whole move operator list and call another move (a new and different type of move) with uniform randomness. We should keep trying to select an alternative move and check whether it has a valid selection of properties. The reallocation move moves an allocation from one house to another. We selected the value of 0.1 to allow a reasonable level of granularity of moves and we choose this value as a compromise between 0 and 1. In Algorithm 5.2, we provide pseudo code outline of our move operator for ITAP locations and bandwidth allocation.

Algorithm 5.2: Generate $(s, \mathcal{M})$

Function *Generate* $(s, \mathcal{M})$*:* $\mathcal{M}$ *list of move operator function*
    Select $m \in M$ with uniform randomness;
    Return $m(s)$
end
Function *Reallocate Placement(s):*
    Select i and j randomly from $\{1, ..., N\}$ such that $i \neq j$;
    Set $s' = s$;
    if $s[i] > 0$*:* **then**
        #Move one ITAP from i to j;
        $s'[i] = s'[i] - 1$;
        $s'[j] = s'[j] + 1$;
        Return s'
    end
    else
    Return Generate $(s, \mathcal{M})$;
end
Function *Reallocate Allocation(s):*
    Select i and j randomly from $\{1, ..., M\}$ such that $i \neq j$;
    Set $s' = s$;
    #Move 0.1 Allocation from i to j;
    $s'[N + i] = s'[N + i] - 0.1$;
    $s'[N + j] = s'[N + j] + 0.1$;
    if $s'[i] < 0$ *or* $s'[j] > 1$*:* **then**
        Return Generate $(s, \mathcal{M})$;
        else
        Return s'
    end
end

Function *Swap Placement(s):*
    Select i and j randomly from $\{1, ..., N\}$ such that $i \neq j$;
    Set $s' = s$;
    if $s'[i] \neq s'[j]$*:* **then**
        #Swap ITAPs between i and j in s';
        $s'[i] = s[j]$;
        $s'[j] = s[i]$;
        Return s'
    end
    else
    Return Generate $(s, \mathcal{M})$;
end
Function *Swap Allocations(s):*
    Select i and j randomly from $\{1, ..., M\}$ such that $i \neq j$;
    Set $s' = s$;
    if $s'[i] \neq s'[j]$*:* **then**
        #Swap Allocation between i and j in s';
        $s'[N + i] = s[N + j]$;
        $s'[N + j] = s[N + i]$;
        Return s'
    end
    else
    Return Generate $(s, \mathcal{M})$;
end

Function *Delete Placement(s):*
  Select i randomly from $\{1, ..., N\}$;
  Set $s' = s$;
  if $s[i] > 0$*:* **then**
    $s'[i] = s[i] - 1$ #Delete one ITAP from i;
    Return s'
  end
  else
  Return Generate $(s, \mathcal{M})$;
end
Function *Delete Allocation(s):*
  Select i randomly from $\{1, ..., M\}$;
  Set $s' = s$;
  $\Delta = random(0, 1)$;
  if $s[N + i] = 0$*:* **then**
    Return Generate $(s, \mathcal{M})$;
  end
  $s'[N + i] = max(s[N + i] - \Delta, 0)$ #Delete random Allocation from i;
  Return s'
end
Function *Delete Delta Allocation(s):*
  if $max(s[N + 1], ...s[N + M]) = 0$*:* **then**
    Return Generate $(s, \mathcal{M})$;
  end
  Set $s' = s$;
  $\Delta = random(0, 1)$;
  for *i in {*$1, ..., M$*}* **do**
    $s'[N + i] = max(s[N + i] - \Delta, 0)$;
  end
  Return s'
end

Function *Add Placement (s):*
    Select i randomly from $\{1, ..., N\}$;
    Set $s' = s$;
    #Add one ITAP to i;
    $s'[i] = s[i] + 1$;
    Return s'
end
Function *Add Allocation(s):*
    Select i randomly from $\{1, ..., M\}$;
    Set $s' = s$;
    $\Delta = random(0, 1)$;
    #Add random Allocation to i
    $s'[N + i] = min(s[N + i] + \Delta, 1)$;
    Return s'
end
Function *Add Delta Allocation(s):*
    #Add random Allocation to *all houses*;
    if $max(s[N + 1], ..., s[N + M]) = 0$*:* **then**
        Return Generate $(s, \mathcal{M})$;
    end
    Set $s' = s$;
    $\Delta = random(0, 1)$;
    for i *in* $\{1, ..., M\}$ **do**
        $s'[N + i] = min(s[N + i] + \Delta, 1)$;
    end
    Return s'
end

5.5 Normalization of the objective functions

Here we define the normalized objective function values. The objectives are normalized approximately to the same units on the same range, as shown in Equations 5.14, 5.15, and 5.16, then multiplied by the appropriate weight and added together to get the total objective value, as defined in the Integer Linear Program Formulation 5.2. The definition of the objective functions, ($f_D(x)$, $f_U(x)$ and $f_I(x)$), separately denotes the normalization of Unserved Demand, Unfairness, and the number of ITAPs, all needing to be minimised.

$$f_D(x) = \frac{\left(\left(\sum_{h=1}^{M} w_h\right) - \left(\sum_{h=1}^{M} \sum_{e=h,v} x_{e,h}\right)\right)}{\sum_{h=1}^{M} w_h} \tag{5.14}$$

$$f_U(x) = \frac{[A - \min_{1 \le h \le M}(\sum_{e=h,v} x_{e,h})]}{A} \tag{5.15}$$

$$f_l(x) = \frac{\sum_{i=1}^{N} y_i \times Cap_i}{\sum_{h=1}^{M} w_h} \tag{5.16}$$

$$A = \min_{1 \le h \le M} w_h \tag{5.17}$$

We calculate the normalized objective values to make them comparable in the weighted function. In Equation 5.14 normalizing unserved demand between 0 and 1, we subtract total flow from total demand and then divide by the total demand. In Equation 5.15 normalizing unfairness, we subtract the minimum flow of all houses from the minimum of all demand and then divide by the minimum of all demand, so that the bandwidth allocation and fairness naturally yields values between 0 and 1. While the ITAPs could be normalized at higher than one, it is hard to normalize ITAPs because there is no natural upper bound on them. We estimate an upper bound by considering the minimum number of ITAPs that would be needed to provide capacity to satisfy all houses. We multiply the total number of ITAPs by their capacity and then divide by the total demand, as shown in Equation 5.16 and 5.17 indicates the minimum of all demand.

Table 5.1: Benchmark data sets

Data Set	No. of Houses	No. of ITAPs	No. of ITAP-Location	Grid Area
DS1	100	10	10	100x100
DS7	1000	50	100	500x500

Table 5.2: Experiments with data sets

Experiments	Data sets	Wireless Range Connectivity	Wireless Link Capacity	ITAP Capacity	Iterations
E5.1	**DS1**	25	5	10	500
E5.2	**DS7**	35	15	20	1000
E5.3	**DS7**	35	54	20	1000

5.6 Experimental Results

To test our algorithms, we used the data set samples shown in Table 5.1. The simulated annealing algorithm was implemented using Python Language 2.7 on a PC with Intel (R) Core (TM) $i7$ $2.40GHz$ and 12 GB RAM. The initial temperature of the SA was set as $T_0 = 2$ with the temperature reduction $\alpha = 0.85$ and the constant $K = 3$. Briefly, to test our approach effectively, we constructed three problem instances by distributing houses and ITAPs in a uniformly random manner across a region, as shown in Table 5.1. The benchmark data sets and Table 5.2 show the implemented experiments. The house capacity was set to 10 Mbps.

To generate an initial solution, 10 ITAPs were placed at randomly chosen locations for $DS1$ and 50 ITAPs were placed randomly for $DS7$. We generated a uniform initial allocation for all houses (of different values, i.e. 0.1, 0.3, 0.5, 0.9, and 1.0). The traffic demand w_h of each house in all experiments was 1. Each experiment was run 15 times with the different random seeds and the average and best results were reported; see Tables 5.3, 5.4 and 5.5.

5.6.1 Weighted sum Performance and Evaluation

We illustrate the numerical results obtained from optimizing multi-objective functions using the weighted-sum approach with the combinations of different move operators in the SA algorithm. Different weight values were applied to all instances with the solutions presented in Tables 5.3, 5.4 and 5.5. The uniform initial allocation for all houses (of different values, i.e. 0.1, 0.3, 0.5, 0.9, and 1.0) shows the spread of solutions on the search space. Allocation is the portion of demand given to each house. The non-dominated sets were calculated using the Non-Dominated Sorting algorithm used with NSGA-II (see Chapter 6). Specifically, the goal of the multi-objective algorithms is to find non-dominated solutions. The concept of domination refers to the situation where two solutions are compared to each other on the basis of whether one solution dominates another solution or not; is a significant concept in MOO. The general definition of domination for multi-objectives can be made by a feasible solution A, which may be said to dominate another feasible solution B, and can be expressed mathematically as:

$A \prec B$, if and only if:

1. The solution A is no worse than B with respect to all objective values, AND
2. The solution A is strictly better than B in at least one objective value;

Therefore solution *A* dominates solution *B*, solution *A* is not dominated by solution *B* or solution *B* is dominated by solution *A* where the notation "$\prec$" represents the domination symbol. Identifying non-dominated solutions in a particular population allows the decision maker to consider a set of solutions that contain the best trade-off. Therefore, for a given set of solutions, making all possible comparisons, will allow us to identify the solutions which belong to the non-dominated set. A Pareto front was deployed to show the enhancement of solutions in the weighted sum. The results are extremely sensitive to the weights used in the weighted sum. In the data sets the experiments runs for 1000 iterations and the tables shows the result of the

initial allocation 0.1. The other initial allocation values are not shown individually but are shown in comparison with other values in Figures 5.4, 5.5 and 5.6. We briefly summarize the experimental results of 3 objective functions with the relative value of Weights of Unserved demand dominant, Unfairness dominant, ITAP dominant, single objective and double objective for DS1, as shown in Table 5.3. The experiment $E5.1$ is repeated with different allocation values. With a low initial allocation we get some unserved demand but the number of ITAPs remains low and fairness values are close together: with high initial allocation everyone is served but with a high number of ITAPs. For example, when the initial allocation is 0.1 then 10% of all demand is served, while when the initial allocation is 1.0 all demands are served but with a high number of ITAPs, as shown below in Figures 5.4, 5.5 and 5.6 with all initial allocations. The non-dominated evaluation solutions of all uniform initial allocation values are displayed as a Pareto optimal front in Figure 5.7.

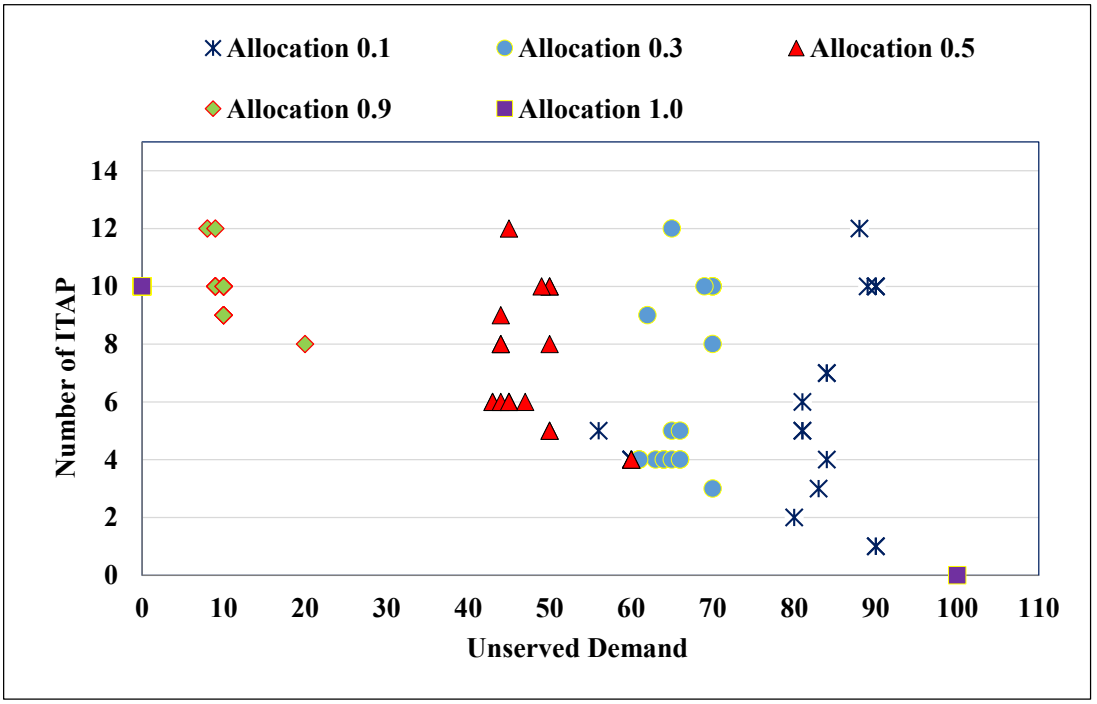

Figure 5.4: Solutions of DS1 with different initial allocation values

The experimental results of 3 objective functions with the relative value of Weights for

Table 5.3: Solutions of 3 objective functions of DS1 for the initial allocation 0.1

Dominants	Unserved Demand Weight	ITAP Weight	Unfairness Weight	Normalized Values			Un Normalized Values		
				f_D	f_I	f_U	f_D	f_I	f_U
Unserved Demand Dominant	0.5	0.1	0.4	0.56	0.5	1	56	5	100
	0.5	0.2	0.3	0.6	0.4	1	60	4	100
	0.5	0.3	0.2	0.6	0.4	1	60	4	100
	0.5	0.4	0.1	0.9	0.1	1	90	1	100
	0.5	0.3	0.15	0.81	0.5	1	81	5	100
	0.34	0.33	0.33	0.6	0.4	1	60	4	100
	0.7	0.07	0.23	0.31	0.7	1	31	7	100
Unfairness Dominant	0.25	0.05	0.7	0.9	1	0.9	90	10	90
	0.32	0.08	0.6	0.8	0.2	1	80	2	100
	0.1	0.4	0.5	0.9	0.1	1	90	1	100
	0.3	0.3	0.4	0.81	0.6	1	81	6	100
	0.2	0.1	0.7	0.9	1	1	90	10	100
	0.23	0.07	0.7	0.89	1	0.9	89	10	90
ITAP Dominant	0.3	0.6	0.1	0.9	0.1	1	90	1	100
	0.1	0.7	0.2	0.9	0.1	1	90	1	100
	0.15	0.8	0.05	0.9	0.1	1	90	1	100
	0.2	0.5	0.3	0.81	0.5	1	81	5	100
Single Objective	1	0	0	0.81	1.8	1	81	18	100
	0	1	0	0.84	0.4	1	84	4	100
	0	0	1	0.9	1	0.9	90	10	90
Double Objective	0.5	0.5	0	0.84	0.7	1	84	7	100
	0	0.5	0.5	0.83	0.3	1	83	3	100
	0.5	0	0.5	0.88	1.2	0.9	88	12	90

DS7 as shown in Table 5.4. The experiment E5.2 is repeated with different allocation values as shown in Figures 5.8, 5.9 and 5.10. The non-dominated evaluation solutions of all the uniform initial allocation values are displayed as a Pareto optimal front in Figure 5.10.

The experimental results of 3 objective functions with the relative value of weights for DS7 of Experiment E5.3 are shown in Table 5.5. The results as shown indicate that increasing wireless capacity will serve the demand from more houses more fairly, which improves the overall fairness of the bandwidth allocation. The experiment is repeated with different allocation values, as shown in Figures 5.12, 5.13 and 5.14. The non-dominated evaluation solutions of all uniform initial allocation values are shown as a Pareto optimal front in Figure 5.15.

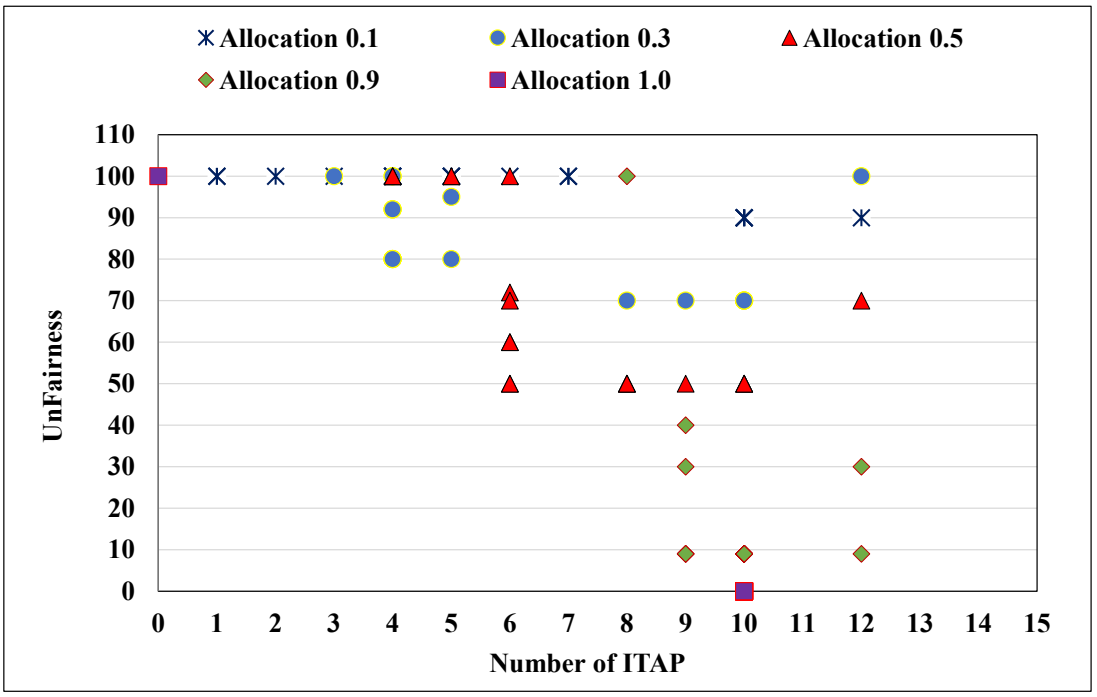

Figure 5.5: Solutions of DS1 with different initial allocation values

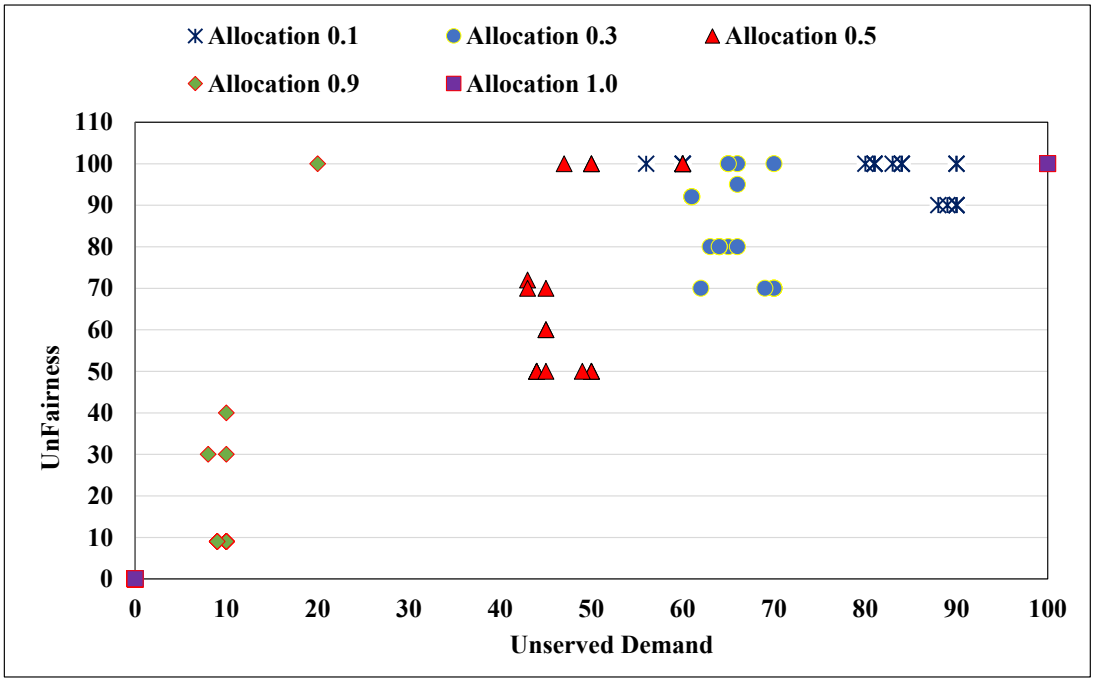

Figure 5.6: Solutions of DS1 with different initial allocation values

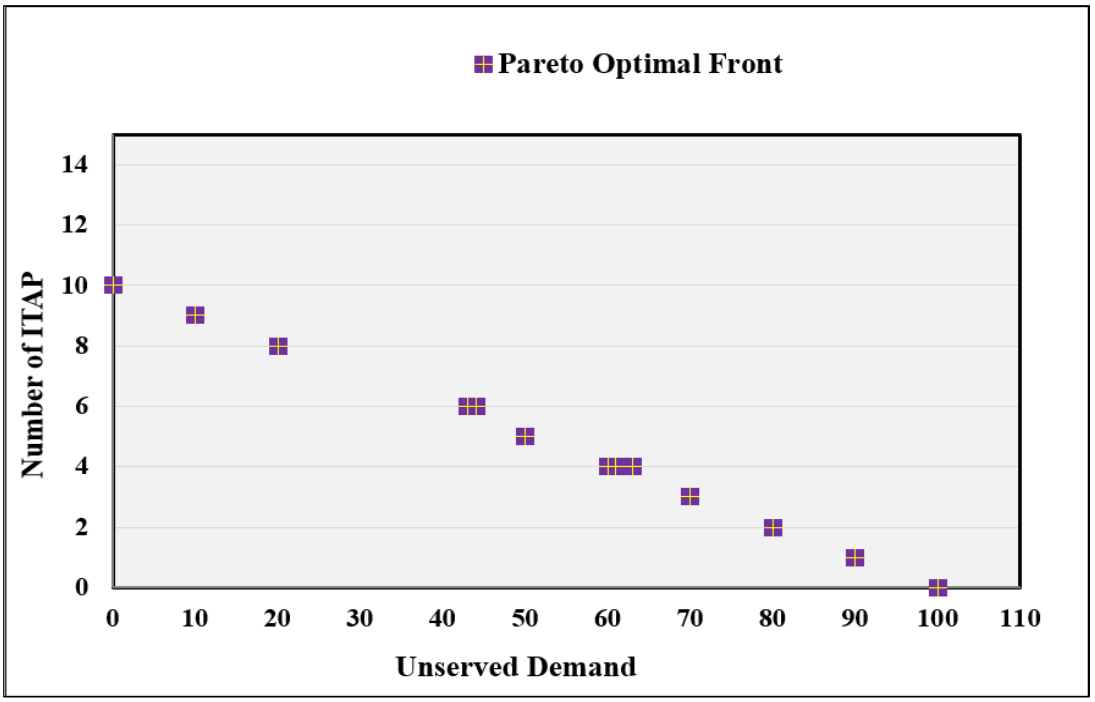

Figure 5.7: Pareto Optimal Front of all the initial Allocations of DS1

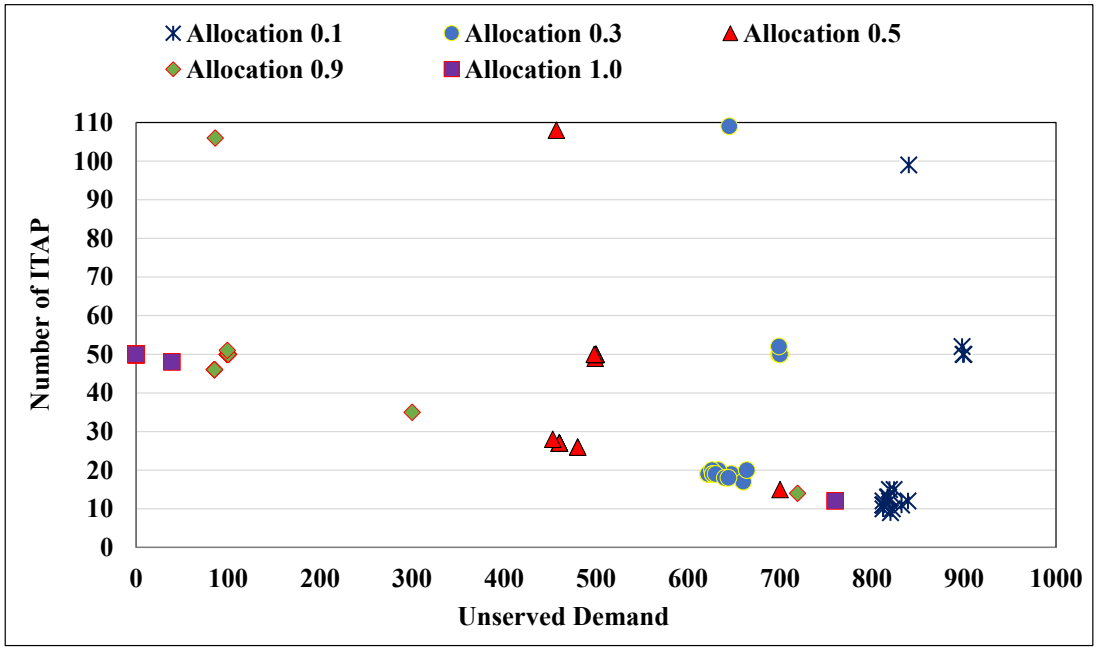

Figure 5.8: Solutions of DS7 of E5.2 with different initial allocation values

Table 5.4: Solutions of 3 objective function of DS7 of E5.2 for the initial allocation 0.1.

Dominants	Unserved Demand Weight	ITAP Weight	Unfairness Weight	Normalized Values			Un Normalized Values		
				f_D	f_I	f_U	f_D	f_I	f_U
Unserved Demand Dominant	0.5	0.1	0.4	0.812	0.24	1	812	12	100
	0.5	0.2	0.3	0.812	0.22	1	812	11	100
	0.5	0.3	0.2	0.813	0.22	1	813	11	100
	0.5	0.4	0.1	0.812	0.22	1	812	11	100
	0.5	0.35	0.15	0.817	0.26	1	817	13	100
	0.34	0.33	0.33	0.812	0.2	1	812	10	100
	0.7	0.07	0.23	0.819	0.28	1	819	14	100
Unfairness Dominant	0.25	0.05	0.7	0.899	1	0.9	899	50	90
	0.32	0.08	0.6	0.819	0.3	1	819	15	100
	0.1	0.4	0.5	0.818	0.26	1	818	13	100
	0.3	0.3	0.4	0.818	0.26	1	818	13	100
	0.2	0.1	0.7	0.82	0.18	1	820	9	100
	0.23	0.07	0.7	0.819	0.24	0.9	819	12	90
ITAP Dominant	0.3	0.6	0.1	0.818	0.26	1	818	13	100
	0.1	0.7	0.2	0.816	0.2	1	816	10	100
	0.15	0.8	0.05	0.82	0.18	1	820	9	100
	0.2	0.5	0.3	0.818	0.26	1	818	13	100
Single Objective	1	0	0	0.84	1.98	1	840	99	100
	0	1	0	0.832	0.22	1	832	11	100
	0	0	1	0.9	1	0.9	900	50	90
Double Objective	0.5	0.5	0	0.824	0.3	1	824	15	100
	0	0.5	0.5	0.839	0.24	1	839	12	100
	0.5	0	0.5	0.898	1.04	0.9	898	52	90

The results of the experiments with the sample of data sets are displayed in Figures 5.4, 5.5, 5.6, 5.8, 5.9, 5.10, 5.12, 5.13 and 5.14; they demonstrate the different solutions of different starting conditions without an aggressive move. The figures indicate that starting with a high allocation value, i.e. 0.9 and 1.0, biases the result towards finding fair solutions. Non-dominated sets of all solutions are provided for comparison. This analysis shows a near linear front for the case without “delta” moves, as seen in Figures 5.7, 5.11 and 5.15.

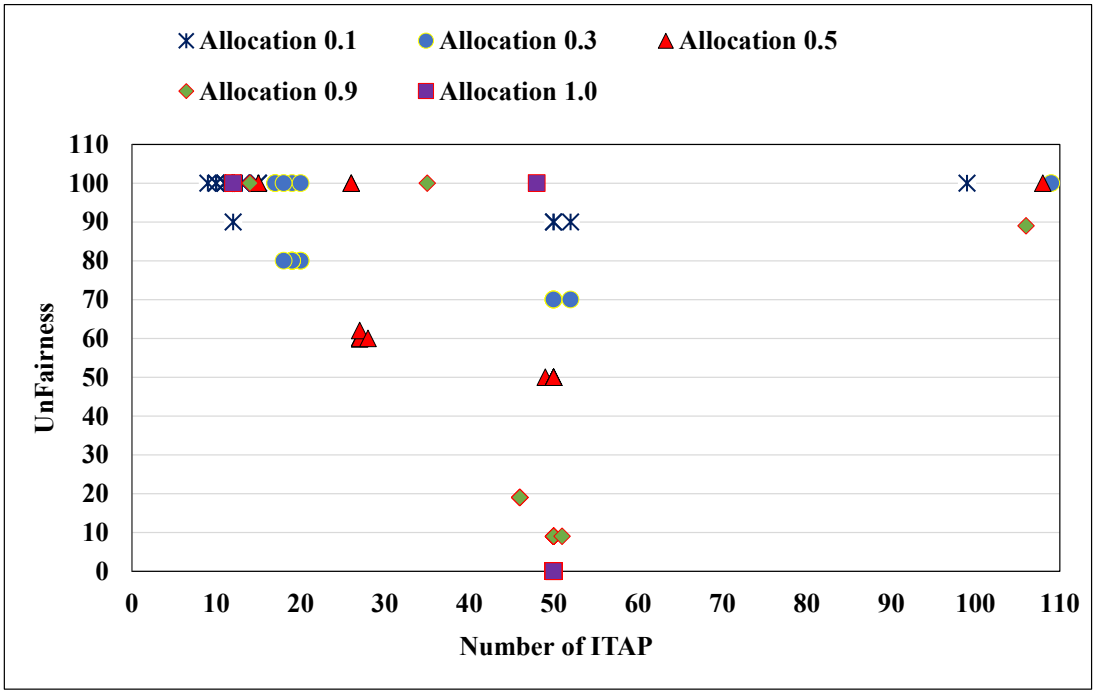

Figure 5.9: Solutions of DS7 of E5.2 with different initial allocation values

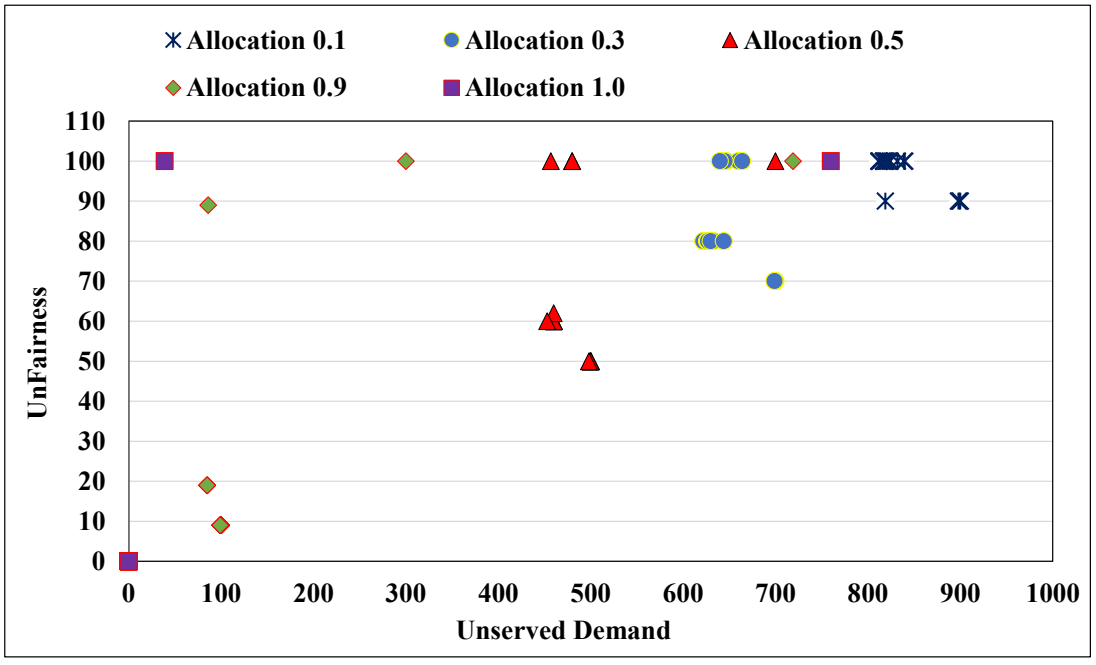

Figure 5.10: Solutions of DS7 of E5.2 with different initial allocation values

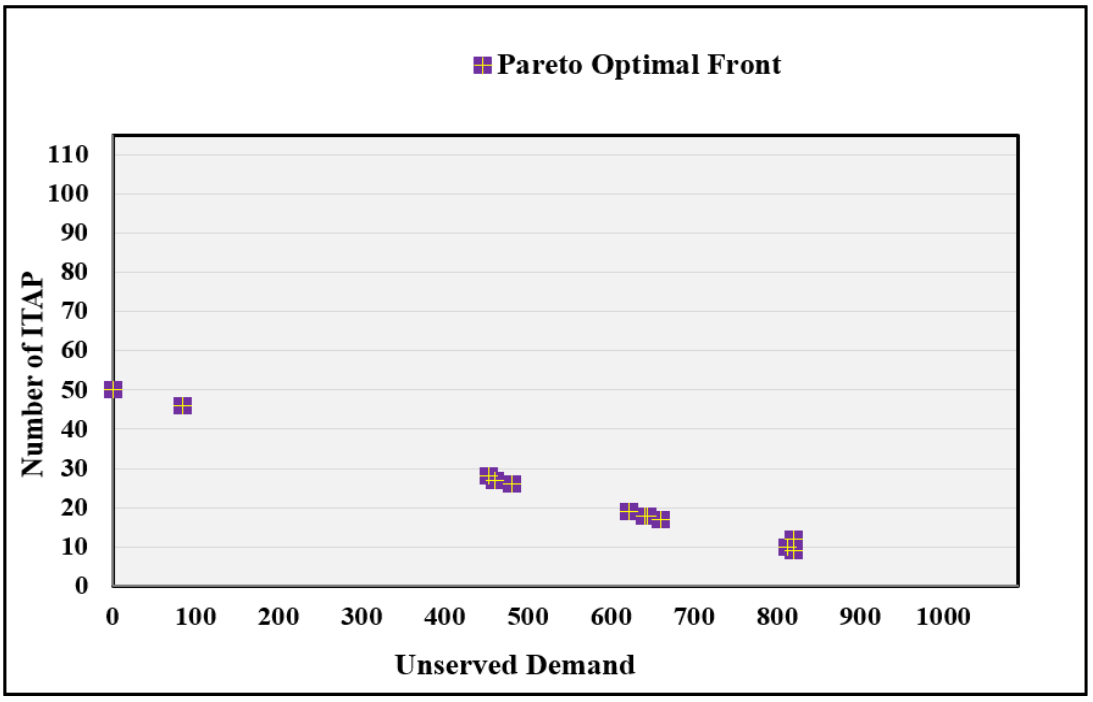

Figure 5.11: Pareto Optimal Front of all the initial Allocations of DS7 of E5.2

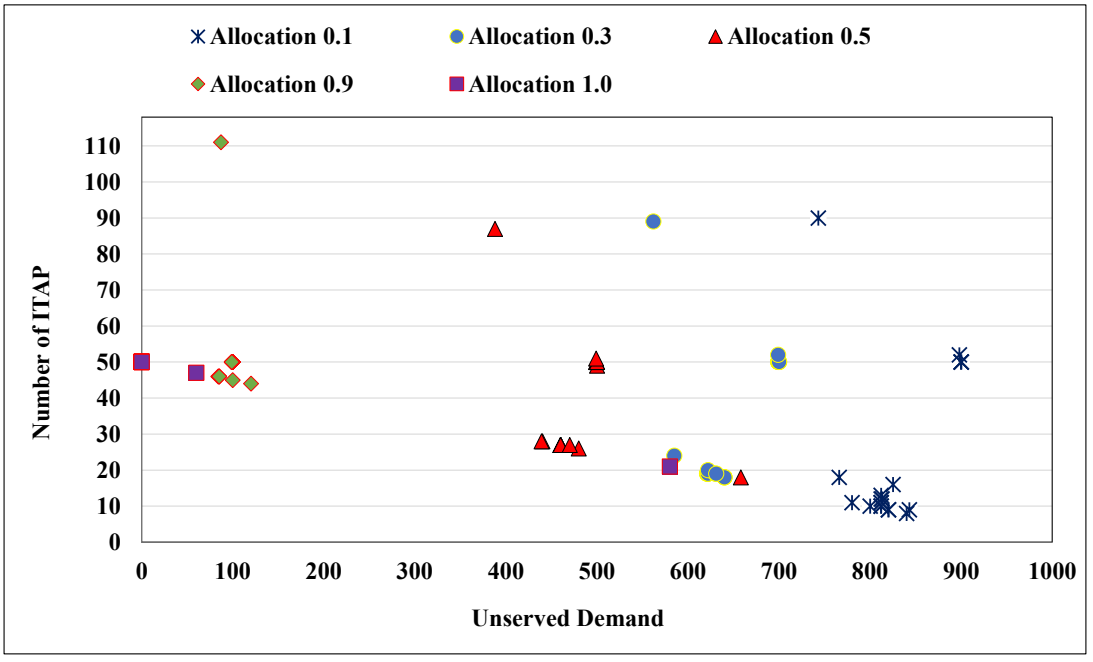

Figure 5.12: Solutions of DS7 of E5.3 with different initial allocation values

Table 5.5: Solutions of 3 objective function of DS7 of E5.3 for the initial allocation 0.1.

Dominants	**Unserved Demand Weight**	**ITAP Weight**	**Unfairness Weight**	**Normalized Values**			**Un Normalized Values**		
				f_D	f_I	f_U	f_D	f_I	f_U
Unserved Demand Dominant	0.5	0.1	0.4	0.812	0.28	1	812	14	100
	0.5	0.2	0.3	0.812	22	1	812	11	100
	0.5	0.3	0.2	0.812	0.22	1	812	11	100
	0.5	0.4	0.1	0.812	0.2	1	812	10	100
	0.5	0.35	0.15	0.812	0.2	1	812	10	100
	0.7	0.07	0.23	0.812	0.24	1	813	12	100
	0.34	0.33	0.33	0.812	0.22	1	812	11	100
Unfairness Dominant	0.25	0.05	0.7	0.9	1	0.9	900	50	90
	0.2	0.1	0.7	0.812	0.24	1	812	12	100
	0.23	0.07	0.7	0.829	0.3	0.9	829	15	90
	0.32	0.08	0.6	0.812	0.28	1	812	14	100
	0.1	0.4	0.5	0.812	0.2	1	812	10	100
	0.3	0.3	0.4	0.812	0.22	1	812	11	100
ITAP Dominant	0.3	0.6	0.1	0.812	0.24	1	812	12	100
	0.1	0.7	0.2	0.824	0.22	1	824	11	100
	0.15	0.8	0.05	0.812	0.24	1	812	12	100
	0.2	0.5	0.3	0.824	0.3	1	824	15	100
Single Objective	1	0	0	0.832	2.28	1	832	114	100
	0	1	0	0.832	0.22	1	832	11	100
	0	0	1	0.9	1	0.9	900	50	0.9
Double Objective	0.5	0.5	0	0.823	0.2	1	823	10	100
	0.5	0	0.5	0.899	1	0.9	899	50	90
	0	0.5	0.5	0.843	0.18	1	843	9	100

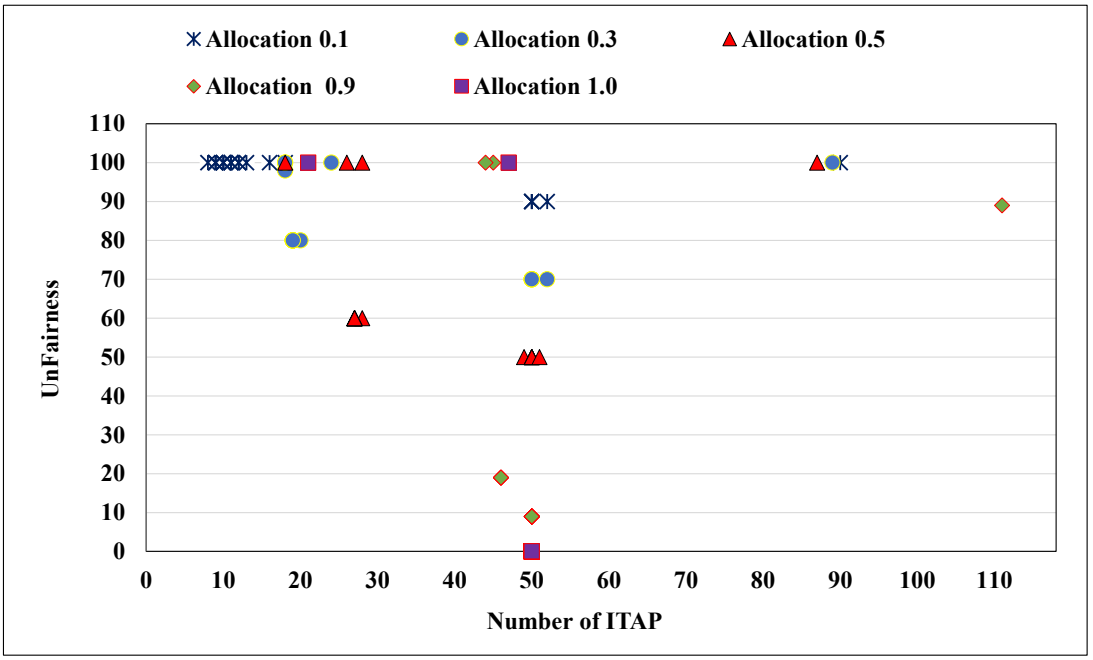

Figure 5.13: Solutions of DS7 of E5.3 with different initial allocation values

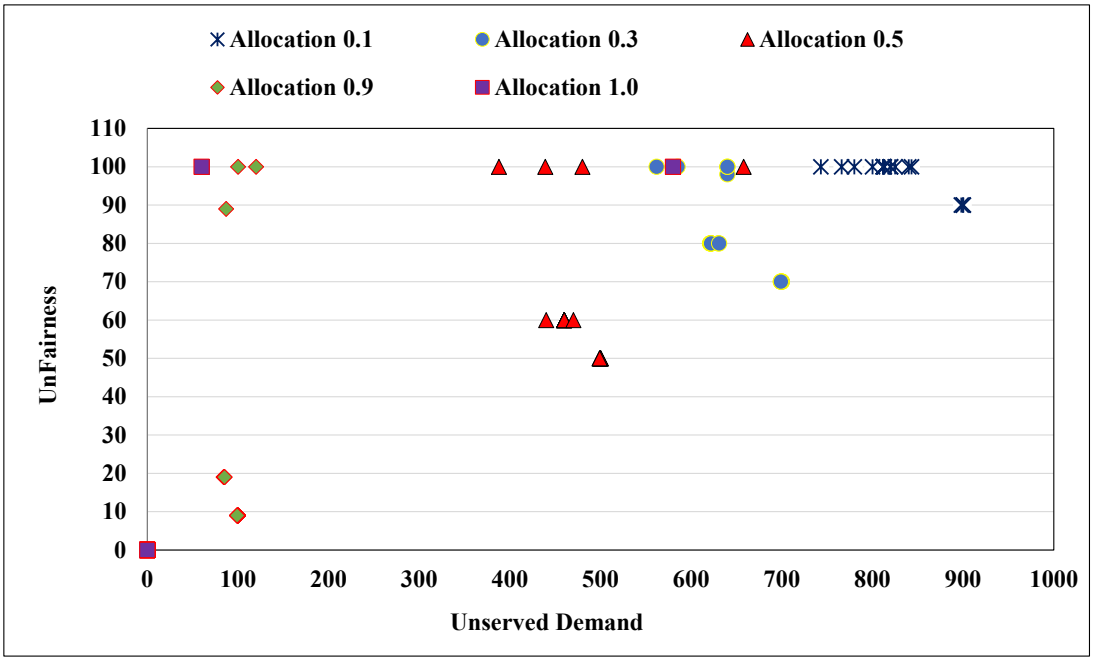

Figure 5.14: Solutions of DS7 of E5.3 with different initial allocation values

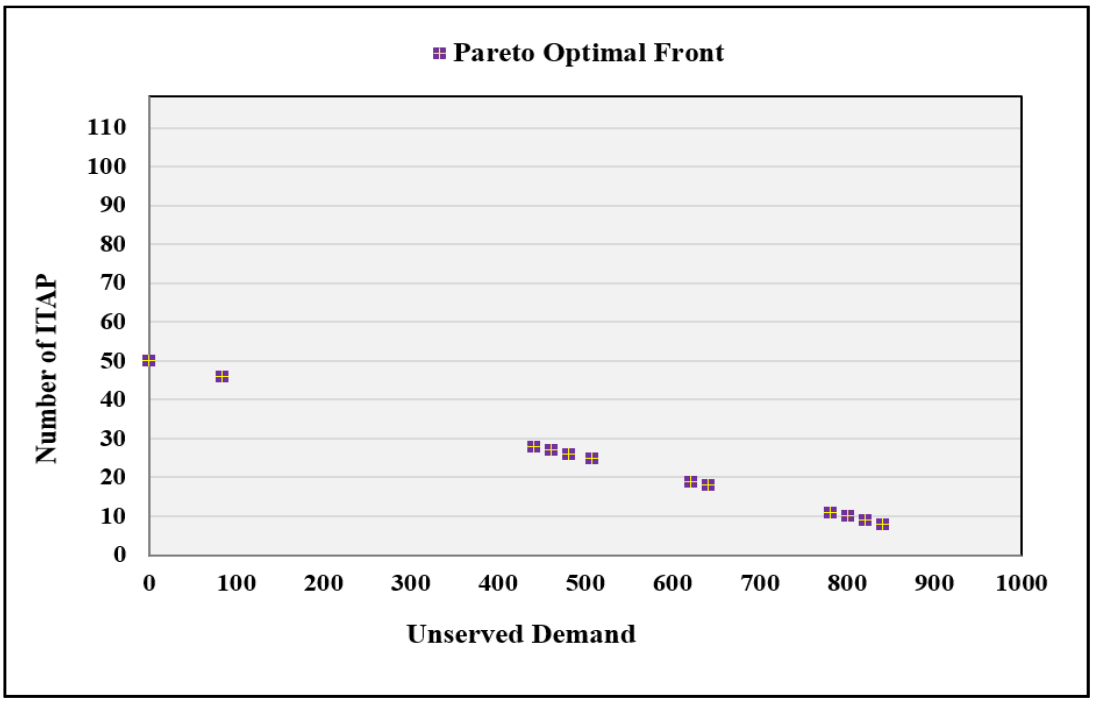

Figure 5.15: Pareto Optimal Front of all the initial Allocations of DS7 of E5.3

5.6.2 Performance and Evaluation of Aggressive moves

We investigate the effect of the more aggressive "delta" moves of adding and deleting allocation, by running each experiment for each initial allocation value with and without the aggressive move. The previous section displayed the results without the aggressive move, but the result of including the "delta" moves showed a degree of improvement in the data sets DS1 and DS7. Because of their diversity it is hard to compare and express the problem of having three objectives. The result shows a diversity and spread of solutions which motivate the use of set coverage. The set coverage metric gives the relative spread of solutions between two non-dominated sets of solutions. Hence the set coverage metric was applied to these results to measure any improvement of solutions with and without the aggressive moves of a uniform initial allocation (over the range $0.1 - 1.0$) as shown in Figures 5.16, 5.17 and 5.18.

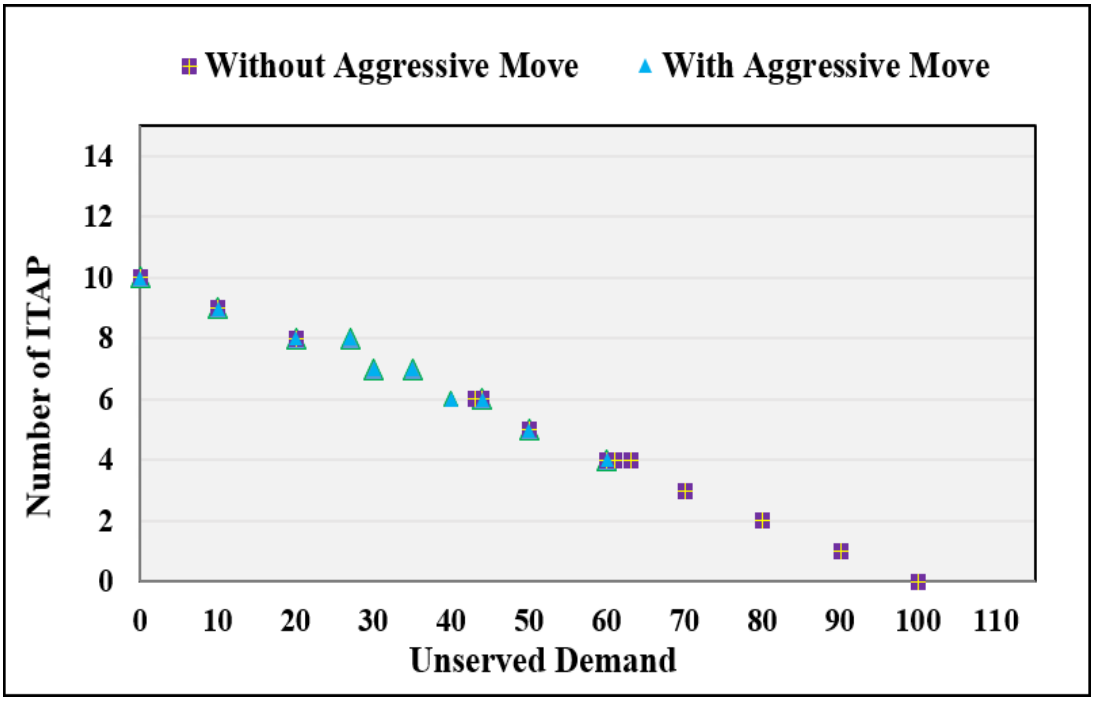

Figure 5.16: DS1of E5.1 with and without the Aggressive Move

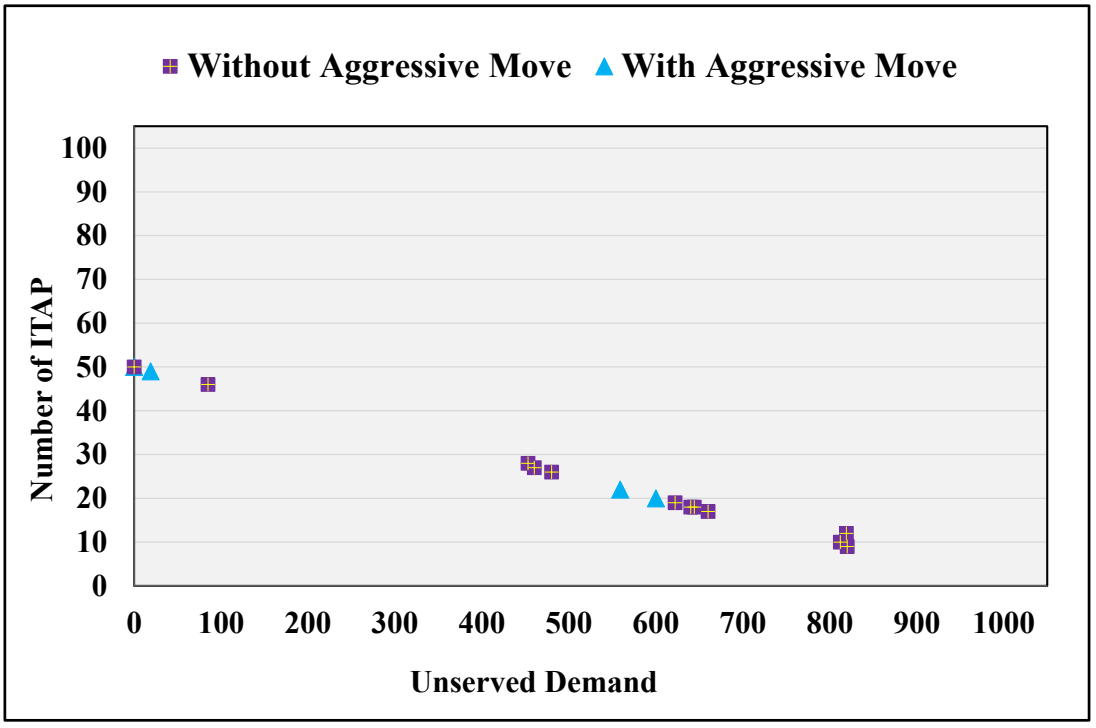

Figure 5.17: DS7 of E5.2 with and without the Aggressive Move

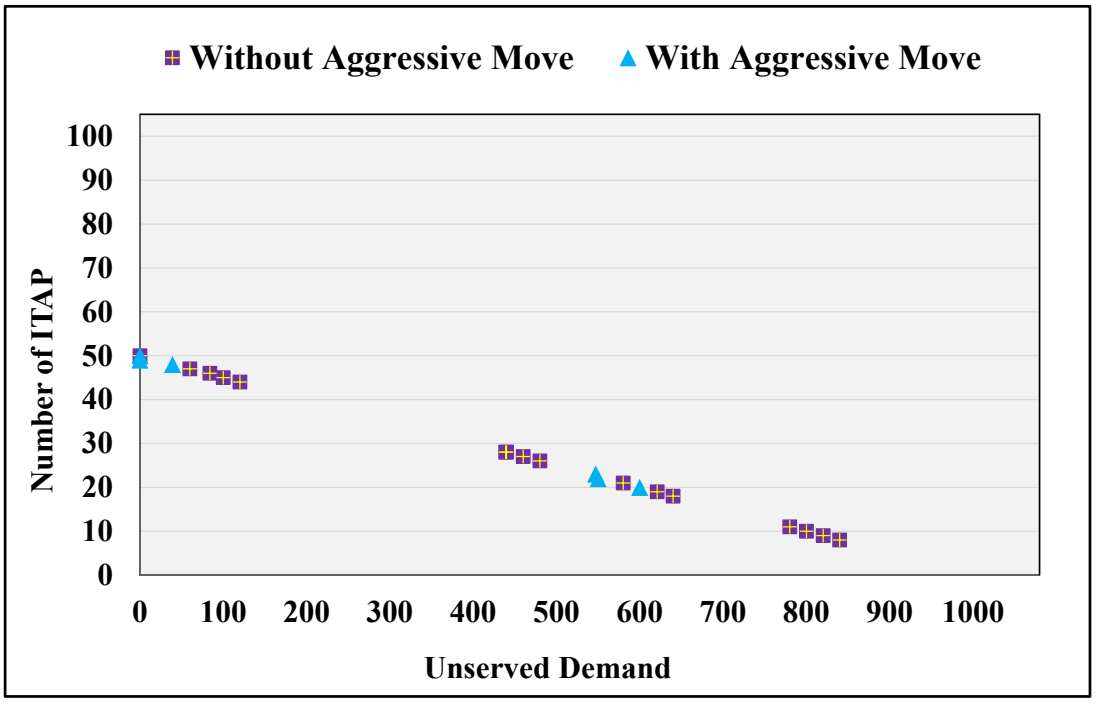

Figure 5.18: DS7 of E5.3 with and without the Aggressive Move

Table 5.6: Solutions of 3 objective functions without and with the Aggressive Move of DS7 of E5.3 for the initial allocation (01).

Dominants	Unserved Demand Weight	ITAP Weight	Unfairness Weight	Un-normalized Values without the Aggressive Move			Un-normalized Values with the Aggressive Move		
				f_D	f_I	f_U	f_D	f_I	f_U
Unserved Demand Dominant	0.5	0.1	0.4	812	14	100	0	50	0
	0.5	0.2	0.3	812	11	100	0	50	0
	0.5	0.3	0.2	812	11	100	0	50	0
	0.5	0.4	0.1	812	10	100	0	50	0
	0.5	0.35	0.15	812	10	100	0	50	0
	0.7	0.07	0.23	813	12	100	0	50	0
	0.34	0.33	0.33	812	11	100	0	50	0
Unfairness Dominant	0.25	0.05	0.7	900	50	90	0	50	0
	0.2	0.1	0.7	812	12	100	0	50	0
	0.23	0.07	0.7	829	15	90	0	50	0
	0.32	0.08	0.6	812	14	100	0	50	0
	0.1	0.4	0.5	812	10	100	0	50	0
	0.3	0.3	0.4	812	11	100	0	50	0
ITAP Dominant	0.3	0.6	0.1	812	12	100	0	50	0
	0.1	0.7	0.2	824	11	100	0	50	0
	0.15	0.8	0.05	812	12	100	0	50	0
	0.2	0.5	0.3	824	15	100	0	50	0
Single Objective	1	0	0	832	114	100	0	51	0
	0	1	0	832	11	100	560	22	100
	0	0	1	900	50	90	0	50	0
Double Objective	0.5	0.5	0	823	10	100	19	49	100
	0.5	0	0.5	899	50	90	0	50	0
	0	0.5	0.5	843	9	100	0	50	0

Table 5.6 show the improvement of aggressive move in the DS7 of E5.3 compared with the solutions without the aggressive move. This demonstrates the balance of the 3 objectives with the range of solutions on the search space (see Figure 5.18).

5.6.3 Set Coverage and Spread

Different metrics could be used in multi-objective optimisation algorithms to compare populations: the two most common performance metrics are set coverage and spread.

In [58] the spread can be calculated based on the separation of non-dominated solutions. The set coverage metric $C(S_A, S_B)$, for each data set for the non-dominated sets of "without aggressive move" and "with aggressive move" are used to show the improvement and to compare the two fronts. Two sets, S_A and S_B, of the non-dominated solutions are defined separately from the "without aggressive move" and "with aggressive move" cases. The set coverage of S_A with respect to S_B, $C(S_A, S_B)$ is the percentage of solutions in S_B, that are weakly dominated by at least one solution in S_A . Hence, if the metric value $C(S_A, S_B) = 100$, it means that for every solution in S_B there is a solution in S_A that is at least as good with respect to all performance measures, and if $C(S_A, S_B) = 0$, it means that no member of S_B is weakly dominated by S_A. That is, the higher the set coverage measure $C(S_A, S_B)$, the greater the number of solutions in S_A that would improve S_B. Since the domination operator is not a symmetric operator, $C(S_A, S_B)$ is not necessarily equal to $1 - (S_B, S_A)$. It is necessary to calculate both $C(S_A, S_B)$ and $C(S_B, S_A)$ to understand how many solutions of S_A are covered by S_B, and vice versa. However, the set coverage of the solutions for "without the Aggressive" and "with the Aggressive" move in $DS1$ of $E5.1$ shows as 50% and 30%, while the set coverage of $DS7$ of $E5.2$ is 8% and 25%, and the set coverage of $DS7$ of $E5.3$ is 25% and 33%. These percentage values illustrate the relative benefit of the aggressive move in $DS1$ and $DS7$. The set coverage results indicate that the Aggressive move has a significant effect on the data sets.

5.7 Chapter Conclusion

This chapter demonstrates that a tool to optimize the multi-objective function of ITAP placement and bandwidth allocation can be implemented using the weighted sum method. In our search for techniques that will yield the best solution, this chapter provides an understanding of the way in which the weighted sum method works on the effect of the move operator in WMNs and explores the significance of the weights with respect to preferences. The weighted sum method provides a simple and user-

friendly approach for multi-objective optimization and shows that there is a trade-off between unfairness and throughput. The result indicates that starting with high allocation value, i.e. 0.9 and 1.0, is biased towards finding fair solutions. Implementing the non-dominated evaluation shows the linear line in the Pareto front of the retrieved solutions achieved by a uniform initial allocation (with different values, i.e. 0.1, 0.3, 0.5, 0.9, and 1.0). The evaluation results show the difficulty of managing the unfairness and sensitivity parameters because no essential guidelines have been presented for selecting weights. A set of weights is considered for emphasizing each of the objectives and we compared the effect of granularity in the moves. The Pareto front results from a specific set of weight values based on our chosen constraints. The evaluation results show that our proposed approach is able to balance the three objectives under review, using the aggressive move operator, and to come up with a range of solutions to cover the search space. The quality of solutions with the aggressive move is better than without the aggressive move because the aggressive move showed a slight improvement in our experiments and it is also sensitive to the weights.

Chapter 6

Multi-Objective Optimization of ITAP Placement using Genetic Algorithms

Evolutionary algorithms (EA) are powerful and popular approaches that are frequently applied to solve many real-world search and optimization problems. EAs are capable of generating good solutions to single-objective problems, but in addition they offer considerable advantages when applied to problems with multiple objectives. In many practical problems, it is common to find that optimizing one objective in a solution leads to a degradation in other objective(s). For example, it clearly possible to increase the overall capacity of a WMN by adding additional ITAPs, but this comes at the expense of additional infrastructure costs. For such problems, it is hard to algorithmically capture which of two solutions is the better overall. For example, consider the solutions to the hypothetical problem shown in Figure 6.1 which has the objectives of minimising both A and B. It is not possible for an optimization process to determine the better solution without additional information from the user - each outperforms the other in one objective. In Chapter 5, this information was provided in the form of a priori weights to encapsulate the relative importance of each, but this approach is limited because the relationships between the objectives are not constant as regards all values of the objectives and it does not admit a diversity of solutions [59]. Moreover, no guidelines are available to show which weight values give the best solutions. The applied weight values need to be decided in advance. Alternatively, EAs address this by maintaining a population of several solutions throughout the optimization process

and allowing the population to have different solutions. Some of these populations are good in cost, while others are good in throughput. Combining the population may have some opportunity to introduce diversity. EAs have the potential of finding multiple Pareto-optimal solutions in a single simulation run for Multi-Objective Optimization (MOO).

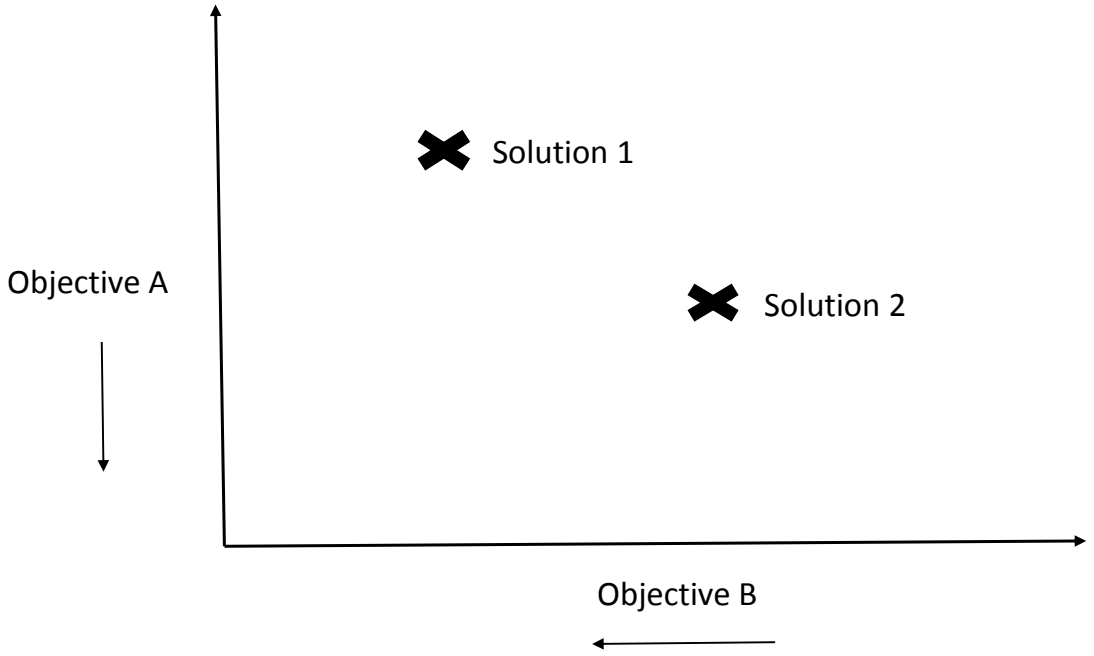

Figure 6.1: A multi-objective optimization problem

As considered in Chapter 5, the problem of optimizing the WMN infrastructure placement is inherently multi-objective, with the aim of minimising the number of ITAPs required, maximising the number of users' demands that can be satisfied and maximising the fairness of bandwidth allocation. In this chapter, a Genetic Algorithm (GA) (a particular class of EA) is applied, specifically the popular Non-Dominated Sorting Genetic Algorithm (NSGA-II) [55] as a potential solution technique for the problem. NSGA-II has been shown to be a widely applicable solution technique and hence the focus is on investigating the definition and application of crossover and mutation operators suitable for this problem. Although GAs are the most frequently encountered type of EA, they are a sub set of EAs. A GA relies on the binary representation of

individuals: an individual is a string of bits, on which the mutation and crossover are easy to implement, whereas an EA relies on customized data structures (instead of a string of bits) and appropriately crafted mutation and crossover operators. A GA or any other EA applies the principles of evolution found in nature to the problem of finding an optimal solution to a solver's problem.

This chapter is organized as follows: Section 6.1 discusses techniques for multi-objective optimization and for defining a series of crossover and mutation operators. Section 6.2 defines the Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) and shows its implementation. Section 6.3 reports the experimental results of using the NSGA-II. Finally, Section 6.4 draws some conclusions.

6.1 Techniques for Multi-Objective Optimization

GAs are very popular meta-heuristic search methods, which are used as the foundation of many evolutionary algorithms to solve optimization problems with single and multiple objectives. In a GA, a population of potential solutions, termed *chromosomes*, is developed over a sequence of generations, using a set of genetic operators called *selection*, *crossover* and *mutation*. Each chromosome represents a solution to the problem and is composed of a list of genes. These are typically bit-strings or vectors of integers/real numbers, but can be more complex, for instance, if they are permutation based. For the problem of optimizing WMN infrastructure placement, we define a chromosome following the same principle as in Chapter 5. That is, for a data set with M houses and N potential ITAP locations, a vector of length $M + N$, is considered, where the first M elements specify the bandwidth allocated to each house and the remaining N elements specify the number of ITAPs to be installed at each location. This can be seen in Figure 6.2. More formally, in a chromosome p:

$p[i]$ is the bandwidth allocation to house i, for $1 \leq i \leq M$

$p[M + i]$ is the number of ITAPs installed at location i, for $1 \leq i \leq N$

For single objective problems, a fitness value can be calculated for each chromosome, whereas for multi-objective problems the fitness value is replaced by a vector of fitness values for each objective. For the problem of optimizing WMN infrastructure placement, the objectives f_D, f_U and f_I as defined in Equations 5.14, 5.15 and 5.16 in Chapter 5 are chosen.

A GA maintains a population of solutions which are updated through a process of reproduction over a number of generations. The overall aim is that the structures in chromosomes that correspond to "good" solutions should survive and propagate into the next generation. Reproduction typically consists of three elements. First, a selection operator is applied to choose the chromosomes that will be part of the reproduction process. At its simplest, pairs of chromosomes are picked from the mating pool at random, but more sophisticated selection operators aim to preferentially choose "good" solutions while avoiding "weak" solutions in the population. For example, tournament selection incorporates elitism where two individuals are randomly picked and the better one is chosen to reproduce once. If they happen to have the same fitness, then one of them is randomly chosen to reproduce. Similarly, roulette wheel selection is performed by picking parents with a probability proportional to their fitness. In NSGA-II, parents are selected from the population using binary tournament selection based on both the quality of their solution and how spread out they are (described in more detail below). In the remainder of the reproduction process, new individuals (termed *children*) are created through the application of *crossover* and mutation operators to a pair of *parent* chromosomes. Crossover operators blend the genetic information between a pair of parent chromosomes to explore the search space, whereas mutation operators are used to maintain sufficient diversity in the population. Crossover allows the basic genetic material of parents to pass to their children, who then form the next generation. A number of crossover operators have been proposed for use in GAs, but in almost all of these, pairs of each gene from the parent chromosomes are combined to pass the corresponding gene on to the child. In what follows the genetic operators are explained in detail. In NSGA-II, *elitism* is used in building the next generation. The elitism op-

erator combines the old population with the newly created population and chooses to keep the better solutions from the combined population. Such an operation makes sure that an algorithm will have a monotonically non-degrading performance. Elitism can speed up the performance of the GA significantly and this can also help to prevent the loss of good solutions once they are found.

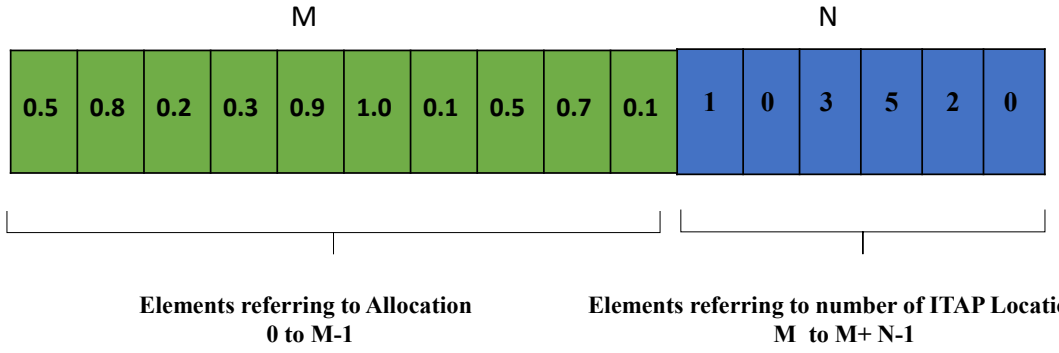

Figure 6.2: Single Chromosome of Allocation and Placement

Many new MOO techniques have been developed, including the Non-dominated Sorting Genetic Algorithm II (NSGA-II). The NSGA-II algorithm sorts the population according to various non-domination levels. The technique of identifying the solutions that belong to a non-dominated set involves comparing all possible pairs of solutions using a dominance operator.

6.1.1 Crossover Operator

This section discusses some of the most commonly used crossover operators. Crossover operators create new offspring by "mating" two selected parents with the aim of maintaining beneficial structures in the children. There are many types of crossover, the most well-known being *single point* crossover. In this simple approach, one crossover point is selected randomly and, to get a new offspring, a binary string from the beginning of the chromosome to the crossover point is copied from one parent and the rest is copied from the other parent. The link order of the genes in the chromosome is important if the ordered chromosomes have a relationship with each other; in this case,

the single point crossover would be suitable for use. However, in this problem the order of genes in the chromosomes is arbitrary and there is no direct relationship between adjacent genes; hence, it is not appropriate to use single point crossover. Selecting and implementing a crossover operator depends on the chromosome representation and also on the optimization problem. Initially two types of crossover operator are applied to WMN: Arithmetic crossover and Uniform crossover

1. **Arithmetic Crossover** Arithmetic crossovers are commonly applied in real-coded GAs; they work by taking the weighted average of the two parents, using the basis of arithmetic mean as shown in Algorithm 6.1. In [60] arithmetic crossover generates a high number of individuals in the search space and creates a greater variety of individuals by increasing the "genetic diversity" of the population while still maintaining adequate coverage of the ranges near and between the parents. The blending of genes, of the two chromosomes from the parents called arithmetic crossover uses the arithmetic mean to produce offspring, as shown in Algorithm 6.1. Arithmetic crossover creates individuals based on the average, seeking to combine elements of the two parents in order to balance their strengths. Previous studies have shown that arithmetic crossover can enhance the rate of convergence [60]. Figure 6.3 shows an example of arithmetic crossover applied to two parents for the ITAP placement problem. As in Chapter 5, $U(s)$ denotes a random value selected as a uniformly random value from the set s.

2. Uniform Crossover Uniform crossover is a simple and often used method for GAs. This idea was first used by Gilbert Syswerda [61], who implicitly assumed that the Bernoulli parameter was $p = 0.5$. For each gene in turn, uniform crossover makes a random, binary decision on which parent to select, based on a specific mixing ratio. For example, with a mixing ratio of 0.5 the child has an equal probability of receiving a given gene from either parent, whereas for a ratio of 0.75, selection is biased towards the first parent. In this thesis, the mixing ratio is fixed at 0.5. This is described formally in Algorithm 6.2, while Figure

Algorithm 6.1: Arithmetic Crossover (p, q, M, N)

p, q: parent chromosomes //(Select p and q randomly from population)

M, N: number of houses, potential ITAP locations

$\sigma = U([0, 1])$

c is an empty chromosome of length $M + N$ *#initialize child to Empty list*

Loop over Houses, build allocations for child from parents

for $i \in \{1, ..., M\}$ **do**

  $c[i] = \sigma.p[i] + (1 - \sigma).q[i]$

end

Loop over Placement, build placement for child from parents

for $i \in \{M + 1, ..., M + N\}$ **do**

  $c[i] = \lfloor \sigma.p[i] + (1 - \sigma).q[i] \rfloor$

end

Return c

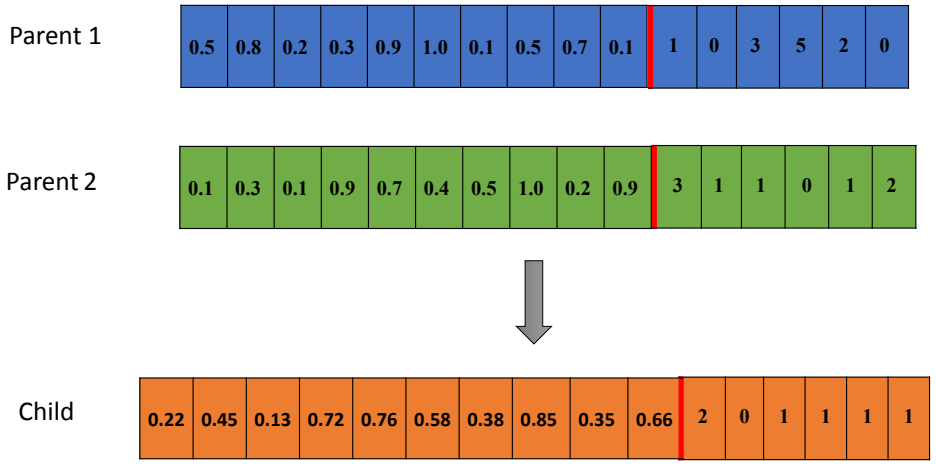

Figure 6.3: Example of Arithmetic Crossover with $\alpha = 0.3$ on individual chromosomes.

6.4 shows an example of uniform crossover.

Algorithm 6.2: Uniform Crossover (p, q, M, N)

p, q: parent chromosomes //(Select p and q randomly from population)
M, N: number of houses, potential ITAP locations
c is an empty chromosome of length $M + N$ // *initialize child to Empty list*
for $i \in \{1, ..., M + N\}$ **do**
    if $U([0, 1]) > 0.5$ **then**
        $c[i] = p[i]$
    end
    else
        $c[i] = q[i]$
    end
end
Return c

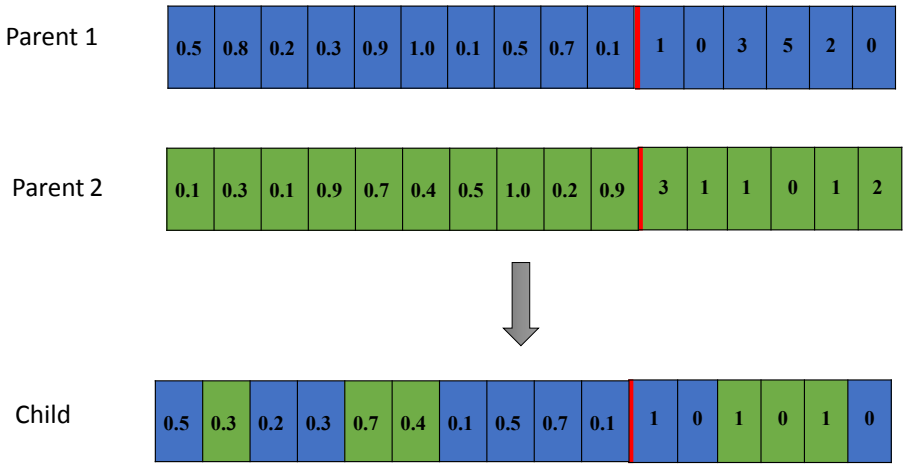

Figure 6.4: Example of Uniform Crossover on Individual Chromosomes

6.1.2 Mutation Operator

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of GA chromosomes to the next. The role of mutation is to introduce diversity into the population to address the local minimum problem, where metaheuristic algorithms get stuck in a local optimum. Mutation is applied to each child with a small probability P_m (termed the *mutation rate*) after crossover has taken place. Mutation alters one or more of the child's gene values in a chromosome from its initial state, to produce diversity from different chromosomes. GA can come to a better solution by using mutation, which occurs during evolution. The best mutation rate is often difficult to determine, since a small value may not introduce sufficient diversity, while a large value leads to many offspring, leading to a random walk in the search space. Two mutation operators GA, *Gaussian* and *Uniform*, are tested here. These types of mutation operator can be used only for integer and real valued genes.

1. Gaussian Mutation

   This operator adds a unit Gaussian distributed random value to the chosen gene of the allocation and placement chromosomes, as shown in Algorithm 6.3. The aim of Gaussian mutation is to avoid being trapped in the local minimum by having more chance to pick genes (a small number of solutions is generated) close to the current solution rather than anywhere else, while mutating the gene preserves more of the current solution [62], and applies diversity on Gaussian principles on a smaller scale. Since allocation values must fall within the range [0, 1], if the mutated value falls outside these limits, it is "clipped" to the maximum/minimum allowed. For the genes corresponding to ITAP placement, clipping is necessary only for the lower bound of 0, but, because Gaussian mutation leads to non-integral values, the resulting values are rounded downwards. The effect of Gaussian mutation is controlled by the standard deviation. The notation of λ_P and λ_A represents the gene mutation probabilities of placement and

allocation. In Gaussian mutation, the ways to mutate genes for placement and allocation are specified below;

- **Each gene corresponding to ITAP placement is modified with probability** λ_P/N

  Adds a unit of Gaussian distributed random value to the original gene.

- **Each gene corresponding to bandwidth allocation is modified with probability** λ_A/M

  With probability λ_A/M, the bandwidth allocated to a house is set by adding a unit of Gaussian distributed random value to the original gene.

P denotes the population, P_m represents the mutation rate, σ_P represent the standard deviation for placement and σ_A represents the standard deviation for the allocation.

2. Uniform Mutation

   Uniform mutation considers each gene in turn and makes a decision whether to modify each gene separately. Normally, this operation replaces the value of the gene with a value selected uniformly randomly between the upper and lower bounds, as described in Algorithm 6.4. However, since there is no upper bound on the number of ITAPs that can be installed at a location, separate gene mutation rates and processes for placement and allocation are applied. For ITAP placement a small perturbation is made to the value, rather than selecting an entirely new value.

   - **Each gene corresponding to ITAP placement is modified with probability** λ_P/N

     Randomly select to either add 1 to or subtract 1 from the original gene.

   - **Each gene corresponding to bandwidth allocation is modified with a given probability.**

Algorithm 6.3: Gaussian Mutate (P, P_m, σ_P, σ_A, λ_P, λ_A, M, N)

p, q: parent chromosomes //(Select p and q randomly from population)
M, N: number of houses, potential ITAP locations
c is an empty chromosome of length $M + N$ *// initialize child to Empty list*
Loop over Houses;
for $i \in \{1, ..., M\}$ **do**
    Checking Mutation rate P_m;
    if $U([0, 1]) < \lambda_A/M$ **then**
        Gaussian mutation & clipping ;
        $c[i] = max(min(c[i] + N(0, \sigma_A), 1.0), 0)$
    end
end
Loop over placement;
for $i \in \{M + 1, ..., M + N\}$ **do**
    Checking Mutation rate P_m;
    if $U([0, 1]) < \lambda_p/N$ **then**
        Gaussian mutation & clipping ;
        $c[i] = max(c[i] + \lfloor N(0, \sigma_P) \rfloor, 0)$
    end
end

Select a random value of delta and then replace the original gene value for each gene of the allocation chromosome. Each of two forms of mutation are selected randomly. First, with the probability λ_A/M, the bandwidth allocated to a house may be set at a random value between 0 and 1. If this change is not made, with probability λ_A/M, the bandwidth allocated to the house is set to 0 (i.e. choosing not to serve them at all).

Algorithm 6.4: Uniform Mutate (P, P_m, λ_P, λ_A M, N)

p, q: parent chromosomes //(Select p and q randomly from population)
M, N: number of houses, potential ITAP locations
c is an empty chromosome of length $M + N$ // *initialize child to Empty list*
Loop over Houses;
for $i \in \{1, ..., M\}$ **do**
    Checking Mutation rate P_m
    if $U([0, 1]) < \lambda_A/M$ **then**
        $c[i] = 0$
    else if $U([0, 1]) < \lambda_A/M$ **then**
        Mutate Gene
        $c[i] = U([0, 1])$
end
Loop over placement;
for $i \in \{M + 1, ..., M + N\}$ **do**
    Checking Mutation rate P_m
    if $U([0, 1]) < \lambda_p/N$ **then**
        Mutate Gene & clipping
        $c[i] = max(c[i] + U(\{-1, 1\}), 0)$
    end
end

6.2 Non-Dominated Sorting Genetic Algorithm-II (NSGA-II)

The Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) was chosen for implementation in the research for solving a multi-objective problem. NSGA-II is a commonly used multi-objective optimization algorithm with three special characteristics

[55]: a) it is a sorting non-dominated procedure where all the individuals are sorted according to the level of non-domination; b) it implements elitism, which stores all non-dominated solutions and hence enhances convergence properties; and c) it adapts a suitable automatic mechanics based on the crowding distance, in order to guarantee diversity and spread of solutions; Constraints are implemented using a modified definition of dominance without the use of penalty functions. The population is initialized to a set of random solutions. At each generation the population is sorted on the basis of non-domination into a number of fronts. The first front, F_0, is the complete non-dominant set in the current population. The second front, F_1, contains only those solutions that are dominated by individuals in the first front, and so on for subsequent fronts (as shown in Figure 6.5). Each individual in a population is assigned a rank (fitness) value based on the front that they belong to (individuals in the first front are given a fitness value of 1 and individuals in the second are assigned a fitness value of 2, and so on). In addition to the fitness value, a new parameter called crowding distance is calculated for each individual. The crowding distance is a measure of how close an individual is to its neighbours in terms of their fitness. Large average crowding distance will result in greater diversity in the population. Parents are selected from the population using binary tournament selection based on rank and crowding distance. An individual is selected if its rank is less than the other's or if the crowding distance is greater than the other's; in other words, crowding distance is compared only if the rank for both individuals is the same. The selected population generates an offspring population Q_0 of size $|P|$ (the same as the number of population) from crossover and mutation operators, as discussed in section 6.1. The population with the current population and current off-springs is sorted again on the basis of non-domination and only the best $|P|$ individuals are selected, by keeping the previous population that got the chance to survive to the next generation if they were better, where $|P|$ is the population size and Q_0 is the initial offspring. The selection is based on rank and on the crowding distance on the last front.

To identify the Pareto set, the procedure of sorting solutions, taken from [55], is im-

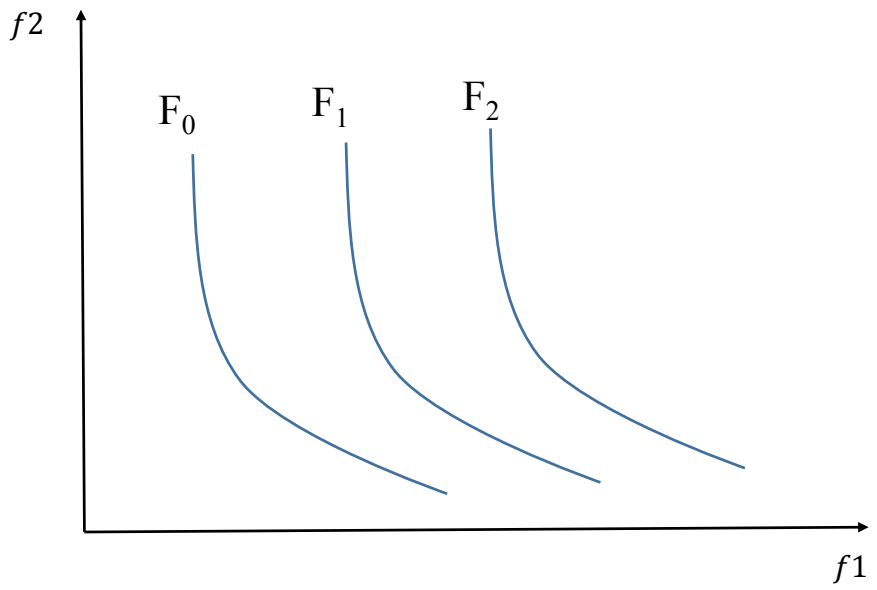

Figure 6.5: Fronts of a non-dominant set in the current population

plemented. In order to identify the non-dominated set, each solution can be compared with every other solution to find if it is dominated. This process continues until all the non-dominated individuals have been identified. For each chromosome in the population P_0 the objectives are evaluated by applying the crowding-distance assignment procedure. Then a fast non-dominated sort is applied to P_0, where P_0 is the initial population and P_t is the population in generation t, (see Algorithm 6.5), which assigns the non-dominated solutions. Initially, for each solution the NSGA-II algorithm calculates two entities:

1) Domination count n_p, the number of solutions which dominate the solution p, and 2) S_p, a set of solutions that the solution p dominates, At each iteration of NSGA-II, start with the population and generate offspring and then combine those offspring with the population ($P_t \cup Q_t$) and sort them out to perform fronts, where the Q_t is the offspring in generation t. Next introduce diversity among the non-dominated solutions by comparing the crowding distance. The Algorithm 6.5 shows the general pseudo code outline of the non-dominated sorting genetic approach [55].

Algorithm 6.5: Fast non-dominated sorting approach (P, I)

```
F1 = { }        Pareto fronts
for each p ∈ P do
    S_p = {}
    n_p = 0
    for each q ∈ P do
        if (p < q)        if p dominates q ;
         then
            S_p = S_p ∪ {q}        Add q to the set of solutions dominated by p ;
        end
        else if (q < p) then
            n_p = n_p + 1        increment the domination counter of p ;
    end
    if n_p = 0        p belongs to first front ;
     then
        F1 = F1 ∪ {p} ;
    end
    Return F1 ;
end
i = 1        Initialize the front counter ;
while F_i ≠ θ do
    Q = θ        Used to store the members of the next front;
    for each p ∈ F_i do
        for each q ∈ S_p do
            n_q = n_q − 1
            if n_q = 0        q belong to the next front;
             then
                q_rank = i + 1
                Q = Q ∪ {q}
            end
        end
    end
end
```

$i = i + 1$;

$F_i = Q$;

$I = |I|$ number of solutions in I ;

for *each I, set* $I[i]_{distance} = 0$ *initialize distance;*

do

  for *each objective* $\mathcal{M}$ **do**

    I = sort(I, $\mathcal{M}$) sort using each objective value;

    $I[i]_{distance} = I[i]_{distance} = \infty$ so that boundary points are always selected;

    for $i = 2\,to\,(I-1)$ *for all other points;*

    do

      $I[i]_{distance} = I[i]_{distance} + (I[i+1].\mathcal{M} - I[i-1].\mathcal{M})/(f_{\mathcal{M}}^{max} - f_{\mathcal{M}}^{min})$

    end

  end

end

$R_t = P_t \cup Q_t$ combine parent and offspring population;

F = fast-non-dominated-sort (R_t) $F = (F_1, F_2, ..)$all non-dominated fronts of R_t

$P_{t+1} = \theta$ and $i = 1$

Until $|P_{t+1}| + |F_i| \leq N$ until the parent population is filled

  Crowding-distance-assignment (F_i) calculate crowding-distance in F_i

  $P_{t+1} = P_{t+1} \cup F_i$ include ith non-dominated front for inclusion

  $i = i + 1$ check the next front for inclusion

Sort ($F_i <_n$) sort in descending order using $<_n$

$P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$ choose the first $(N - |P_{t+1}|)$ elements of F_i

Q_{t+1} = make-new-pop (P_{t+1}) use selection, crossover and mutation to create a new population Q_{t+1}

$t = t + 1$ increment the generation counter

Table 6.1: Benchmark data sets

Data Set	No. of Houses	No. of ITAPs	No. of ITAP-Location	Grid Area
DS1	100	10	10	100×100
DS1A	100	10	10	100×100
DS7	1000	50	100	500×500
DS7A	1000	50	100	500×500

6.3 Experimental Results

NSGA-II algorithm 6.5 was run for the parameters below to test the performance of different combinations of crossover and mutation operators for the data sets described in Table 6.1. Each of the following experiments was run 5 times on each test case with five different random seeds. The results are presented for the mean values of the 5 runs. The experimental results are discussed in the following sections.

6.3.1 Parameter Settings

The arithmetic and uniform crossover with uniform and Gaussian mutation were implemented and tested for different combinations of population size and generations with a population ranging from 16 to 200 and a generational range from 125 to 500. A population size of 32 and max generation of 500 are chosen, with a mutation rate (P_m) of 0.1 and gene mutation probabilities of $\lambda_P = \lambda_A = 1$. The data sets in Table 6.1 were applied for the wide range of instances for the experiments in Table 6.2. DS1 and DS7 were regenerated with different random seeds for sets of ITAP and house locations to generate new data sets with the same density, i.e. DS1A and DS7A, with the properties as shown in Table 6.1.

Some initial experiments were performed to determine a population size and number of generations that would be appropriate, bearing in mind that a smaller population size leads to quicker convergence but the algorithm is more likely to get trapped in local

Table 6.2: Experiments on data sets

Experiments	Data sets	Wireless Range Connectivity	Wireless Link Capacity	ITAP Capacity	Generation
E6.1	**DS1**	25	5	10	500
E6.2	**DS1A**	25	5	10	500
E6.3	**DS7**	35	15	20	500
E6.4	**DS7A**	35	15	20	500
E6.5	**DS7A**	35	15	20	1000
E6.6	**DS7**	35	54	20	500

Table 6.3: The parameters of the experiments

Parameter	P_m	σ_P	σ_A	λ_P	λ_A	**House demand**
Value	0.1	1	1/6	1	1	1

optima and conversely that a large population size will affect the ability of the GA to explore the whole search space equally. In [63], it is shown how increasing the population size also increases the structural bias of the GA. This means that the search is biased towards specific regions of the search space while others are ignored. However, in light of the above facts and the sensitivity analysis of the results the population size was reduced and run for a longer time, for more generations. To see the progress of the algorithm; a population of 32 individuals and 500 generations were applied as a reasonable balance between quality and runtime in the final solution.

6.3.2 Crossover Operator Experimentation

The first experiments aimed to investigate the effectiveness of the crossover operator. Both arithmetic and uniform crossover were applied with Gaussian mutation operators using the parameters in Table 6.3.

5.14 To compare the final populations produced by each crossover, the set coverage metric as described in section 5.6 in Chapter 5 was applied. Five runs with different random seeds were generated for each crossover type to give sets of populations A and

Table 6.4: Set coverage of Arithmetic and Uniform crossover with Gaussian mutation for DS1 and DS7.

Set Coverage B	**Set Coverage A**									
Gaussian	**Gaussian Mutation / Arithmetic Crossover**									
Mutation/	**E6.1**		**E6.2**		**E6.3**		**E6.4**		**E6.6**	
Uniform	**(B, A)**	**(A, B)**	**(B, A)**	**(A, B)**	**(B, A)**	**(A, B)**	**(B, A)**	**(A, B)**	**(B, A)**	**(A, B)**
Crossover	46%	20%	73%	16%	6%	57%	14%	48%	7%	61%

B and both sets were compared pairwise:

$A = \{A_1, A_2, A_3, A_4, A_5\}$

$B = \{B_1, B_2, B_3, B_4, B_5\}$

$S = \frac{1}{25} \sum_{1 \leq i,j \leq 5} S(Ai, Bj)$

The average of set coverage is shown in Table 6.4, indicating uniform crossover outperformed the arithmetic crossover in DS1 and DS1A of E6.1 and E6.2, and the arithmetic crossover outperformed the uniform crossover in DS7 and DS7A of E6.3, E6.4 and E6.6. The assumption is that uniform crossover does better in small data sets, which may be easy problems, whereas arithmetic does better in larger data sets because the problem then is harder.

The initial result showed that uniform did better in DS1 and DS1A; as suspected, because of the size of the problem. To investigate this the progression of uniform and arithmetic crossovers was examined through the generations for data sets DS1A. In Table 6.5, the set coverage after 100 generations shows that the arithmetic and uniform are roughly the same but that after 200 generations the uniform crossover does much better than the arithmetic crossover. After 300 generations, however, the arithmetic does better than the uniform. After generations 400 and 500 uniform crossovers show much divergence. The evaluation result is that the set coverage varies from 100 generations to 500 generations for both arithmetic and uniform crossovers, as shown in Figure 6.6. Effectively, with simulating smaller problems it is easy to achieve improvement and easy to explore the search space but the visible effect is small. Given the suspicion

Table 6.5: Set coverage of Arithmetic and Uniform crossover with Gaussian mutation for generations (100 - 500) of DS1A.

Set Coverage B Gaussian Mutation/ Uniform Crossover	**Set Coverage A Gaussian Mutation/Arithmetic Crossover E6.2**									
	100 Generations		**200 Generations**		**300 Generations**		**400 Generations**		**500 Generations**	
	(B, A)	**(A, B)**	**(B, A)**	**(A, B)**	**(B, A)**	**(A, B)**	**(B, A)**	**(A, B)**	**(B, A)**	**(A, B)**
	41%	38%	91%	14%	27%	33%	88%	27%	93%	15%

that a small data set denotes an easy problem, it was proposed to investigate harder problems.

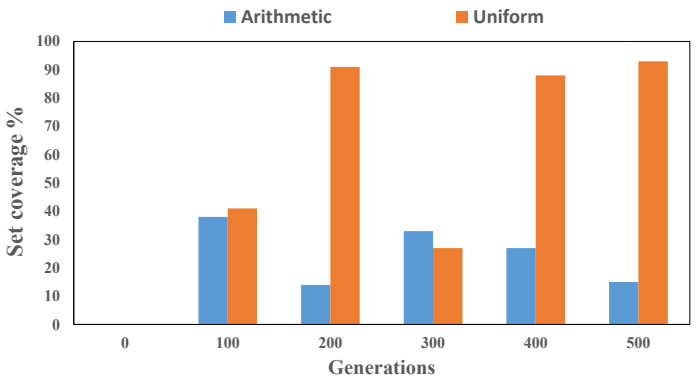

Figure 6.6: Set coverage of Arithmetic and Uniform Crossover of DS1

To confirm and draw attention to the performance of arithmetic crossovers in comparison to uniform crossovers in DS7 and DS7A, the NSGA-II algorithm was run once with 1000 generations for DS7A, to compare uniform and arithmetic crossovers. The longer the experiment ran, the greater the difference found between the two types of crossover. As before, set coverage was compared after every 100 generations. As seen in Table 6.6, uniform crossover initially outperforms arithmetic, but arithmetic

Table 6.6: Set coverage of Arithmetic and Uniform crossover with Gaussian mutation of 1000 generations of DS7A.

Set Coverage B Gaussian Mutation/ Uniform Crossover	Set Coverage A Gaussian Mutation/Arithmetic Crossover E6.5									
	100 Generations		200 Generations		300 Generations		400 Generations		500 Generations	
	(B, A)	(A, B)	(B, A)	(A, B)	(B, A)	(A, B)	(B, A)	(A, B)	(B, A)	(A, B)
	10%	0%	4%	29%	5%	62%	5%	59%	10%	58%
	600 Generations		700 Generations		800 Generations		900 Generations		1000 Generations	
	5%	65%	9%	56%	5%	62%	0%	76%	4%	56%

does better with the remaining generations. This indicates the good progress in larger problems of arithmetic crossover compared to uniform crossover and confirms that arithmetic crossover is more effective with larger data sets.

In [60] the evaluation results show that algorithms which use the arithmetic crossover consistently outperform those using the uniform crossover; the arithmetic crossover is consistently able to reach the neighbourhood of global minima with competitive speeds of convergence. It is clear from the above evaluation result that, while NSGA-II with arithmetic crossover had the highest average set coverage in all test case of DS7 and DS7A, uniform crossover performed well in small data sets such as DS1and DS1A. Hence, further investigation was suggested into the use of uniform crossover in small data sets and arithmetic crossover in larger data sets. From the above experiments where arithmetic and uniform crossover were tested, the focus was determined as uniform crossover in the house allocation, to improve allocation and serve more demands, since arithmetic crossover would pull everyone towards the middle, because arithmetic crossover cannot give anyone either a full allocation or no allocation at all. In other words, it would be hard to make progress in this direction. For the ITAP placement, arithmetic crossover was focused on improving the attainment of this objective, as shown in Tables 6.7 and 6.8.

Table 6.7: The non-dominated solution of uniform crossover for DS7

Uniform Crossover		
f_D	f_U	f_I
480	0.984649016	26
720	0.959059339	14
760	0.97075792	12
580	0.973183139	21
540	0.982669909	23
573.2467573	0.97075792	23
451.4899448	0.982715658	29
640	0.959059339	18
440	0.988399746	28
680	0.959059339	16
500	0.977933187	25
560	0.975285557	22
600	0.971794411	20
608.7278695	0.959059339	23
740	0.963675021	13
483.8021747	0.977933187	26
556.7327874	0.981447794	23
398.6248681	0.988399746	32
620	0.959059339	19
700	0.959059339	15
460	0.978863435	27
660	0.959059339	17
520	0.999820396	24
423.9703524	0.984649016	30

Table 6.8: The non-dominated solution of arithmetic crossover for DS7

Arithmetic Crossover		
f_D	f_U	f_I
860	0.848318	7
640	0.957089	18
497.5753	0.840982	44
540	0.895586	23
560	0.979841	22
600	0.9456	20
720	0.890432	14
760	0.857483	12
680	0.85879	16
800	0.918517	10
660	0.867393	17
740	0.864393	13
700	0.923822	15
503.1	0.994215	25
489.1366	0.990419	27
780	0.986285	11
580	0.860906	21
520	0.864764	24
840	0.83976	8
820	0.950968	9

A simple example is shown in Figure 6.7, which illustrates the techniques of uniform and arithmetic crossover. It is clear that uniform crossover has a chance of selecting a bad solution or an empty gene, while arithmetic crossover tends to move away from selecting a bad solution. This is an advantage, but its downside is that it also tends to move away from good solutions. The arithmetic crossover takes the average of the

parents so as to strike a balance between their strengths which smooths it out; this is a better method than taking one another, as in uniform crossover.

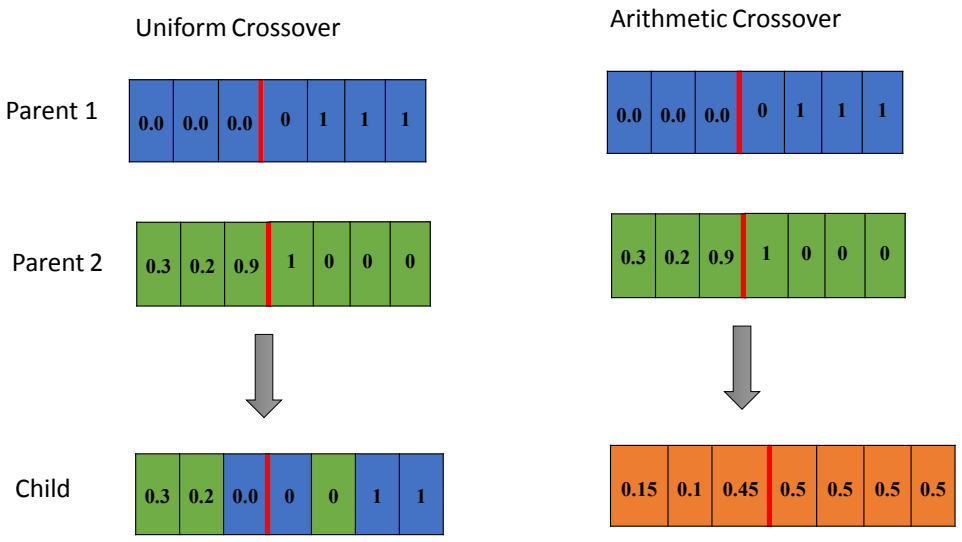

Figure 6.7: Show the convergence of Uniform and Arithmetic crossover with $\alpha = 0.5$.

As expected from the previous result, it was necessary to combine the two crossovers and introduce the *Hybrid* crossover, described below. The hybrid crossover can be represented by setting a mixture of the uniform crossover for allocation genes and the arithmetic crossover for placement genes; this increases the genetic diversity (number of different gene combinations) within a species, with the effect of improving the genetic characteristics of the offspring. A *hybrid* crossover is proposed, which takes two parents and applies a mixture of uniform crossover (where it uses a fixed mixing ratio of 0.5) for the genes relating to allocation and arithmetic crossover for the genes relating to ITAP placement, as shown in Algorithm 6.6. Mutation was then applied to the child as before. Figure 6.8 shows an example of hybrid crossover applied to two parents.

Next the hybrid crossover was applied with Gaussian mutation using the parameters shown in Table 6.9 as before and its set coverage results were compared with the arith-

Algorithm 6.6: Hybrid Crossover (p, q, M, N)

p, q: parent chromosomes //(Select p and q randomly from population)
M, N: number of houses, potential ITAP locations
$\alpha = U([0, 1])$
c is an empty chromosome of length $M + N$ *// initialize child to Empty list*
Loop over Houses - build allocations for child from parents
for $i \in \{1, ..., M\}$ **do**
    if $U([0, 1]) > 0.5$ **then**
        $c[i] = p[i]$
    end
    else
        $c[i] = q[i]$
    end
end
Loop over placement, build placements for child from parents
for $i \in \{M + 1, ..., M + N\}$ **do**
    $c[i] = \lfloor \alpha.p[i] + (1 - \alpha).q[i] \rfloor$
end
Return c

Table 6.9: The parameters of the experiments

Parameter	P_m	σ_P	σ_A	λ_P	λ_A	**House demand**
Value	0.1	1	1/6	1	1	1

metic and uniform crossover.

Five runs with different random seeds were generated for each crossover type to give sets of populations A and B, as described earlier in this section. The average set coverage for the small data (DS1 and DS1A) in Table 6.10 shows the performance of the uniform crossover compared to the arithmetic and hybrid crossovers. The arithmetic

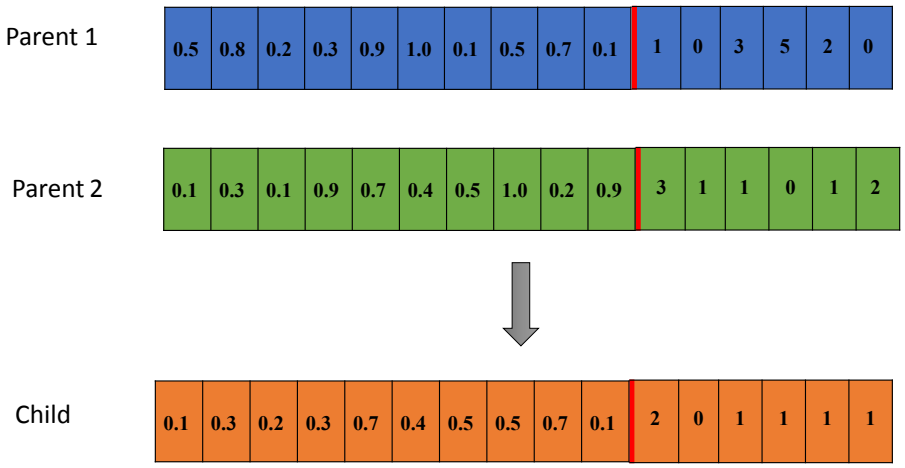

Figure 6.8: Hybrid Crossover with $\alpha = 0.3$ on individual chromosomes

Table 6.10: Set coverage of Arithmetic, Uniform and Hybrid crossover with Gaussian mutation of DS1 and DS1A.

Set Coverage B Gaussian Mutation	**Set Coverage A Gaussian Mutation**							
	E6.1				**E6.2**			
	Hybrid		**Arithmetic**		**Hybrid**		**Arithmetic**	
	(B, A)	**(A, B)**	**(B, A)**	**(A, B)**	**(B, A)**	**(A, B)**	**(A, B)**	**(A, B)**
Uniform Crossover	45%	27%	46%	20%	48%	26%	73%	16%
Arithmetic Crossover	35%	29%			23%	41%		

and hybrid were more or less close to each other in E6.1 while in E6.2 the hybrid outperformed the arithmetic.

The set coverage for the larger data sets (DS7 and DS7A) in Table 6.11 demonstrates the performance of arithmetic crossover compared to that of the uniform and hybrid crossover.

Figure 6.9 shows the spread of solutions for data set DS7 for arithmetic, uniform and hybrid crossovers. The non-dominated solutions are located on the Pareto fronts. As

Table 6.11: Set coverage of Arithmetic, Uniform and Hybrid crossover with Gaussian mutation of DS7 and DS7A.

Set Coverage B Gaussian Mutation	Set Coverage A Gaussian Mutation											
	E6.3				E6.4				E6.6			
	Hybrid		Arithmetic		Hybrid		Arithmetic		Hybrid		Arithmetic	
	(B, A)	(A, B)	(B, A)	(A, B)	(B, A)	(A, B)	(B, A)	(A, B)	(B, A)	(A, B)	(B,A)	(A, B)
Uniform Crossover	14%	46%	6%	57%	33%	14%	14%	48%	25%	30%	7%	61%
Arithmetic Crossover	57%	7%			45%	16%			58%	8%		

the algorithm progresses, the graphs show the population after each 100 generations, which are represented with different colours. They show the rapid convergence of arithmetic crossover with Gaussian mutation compared to the more gradual convergence of uniform crossover with Gaussian mutation. All solutions came out with zero unfairness.

The evaluation results indicate that uniform crossover worked best on small data sets and arithmetic crossover had the best average set coverage for larger data sets; therefore arithmetic crossover and uniform crossover were subjected to further investigation.

6.3.3 Mutation Operator Experimentation

To compare the performance of the proposed mutation operators, experiments were performed on the data sets of Table 6.1 to explore the efficiency of mutation when paired with uniform crossover for DS1 and arithmetic crossover for DS7, using the parameters shown in Table 6.12 , as before. The set coverage results for these experiments are shown in Table 6.13, which illustrates that Gaussian mutation gives better results than uniform mutation.

Table 6.13 shows that Gaussian mutation has a slight tendency to outperform uniform mutation in both small and large data sets. Hence Gaussian mutations are the most

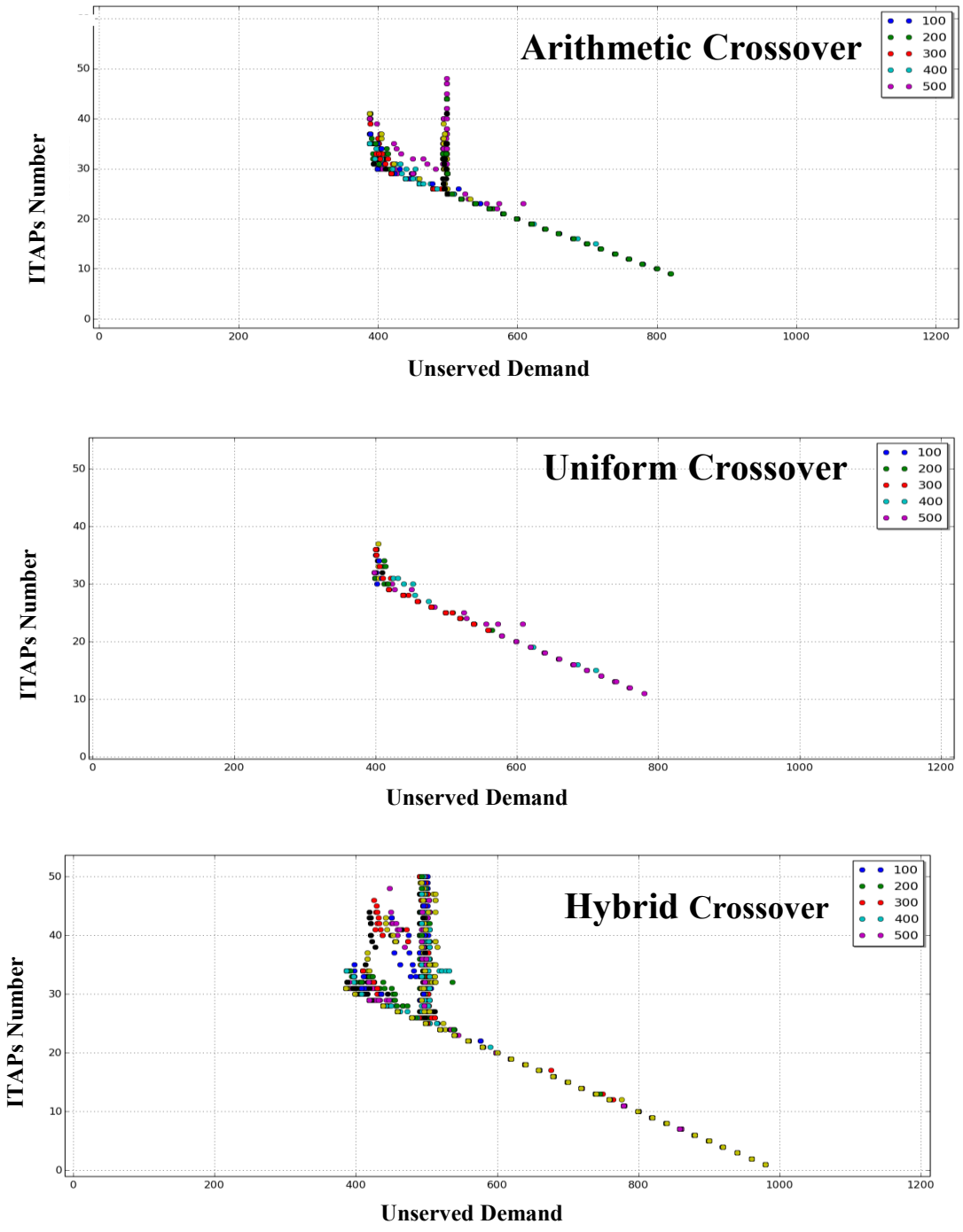

Figure 6.9: Solutions of Arithmetic, Uniform and Hybrid Crossover for DS7

effective mutation operators in the research; therefore, they should be considered for further investigation.

Table 6.12: The parameters of the experiments

Parameter	P_m	σ_P	σ_A	λ_P	λ_A	**House demand**
Value	0.1	1	1/6	1	1	1

Table 6.13: Set coverage of Arithmetic crossover with Uniform and Gaussian Mutation of DS1, DS2 and DS3.

Set Coverage B Uniform Mutation	**Set Coverage A Gaussian Mutation**	
	(B, A)	**(A, B)**
DS1 and DS1A	38%	40%
DS7 and DS7A	27%	30%
DS7 - E6.6	15%	40%

6.3.4 Lifting Allocation Mutation

The mutation operators defined and applied so far are limited in the extent of the changes that they make. For small values of P_m, λ_P and λ_A, there is very little perturbation to the chromosome, which is unlikely to have a significant effect on the overall cost. However, larger values of P_m, λ_P and λ_A perturb the chromosome in an uncoordinated fashion, potentially losing any beneficial structure in the solution. To address this, a modified mutation operator was proposed that applies changes to all the bandwidths allocated to all the houses in a coordinated fashion, adding the same random value to them all. *Lifting allocation mutation* is performed by adding a single normally distributed random value (delta) to every house, as described in Algorithm 6.7. Lifting is performed with probability P_r, otherwise the previously defined Gaussian mutation is applied. The P_r represents the probability rates of 0.3, 0.5 and 0.9.

The lifting mutation within NSGA-II was applied to the data sets. The set coverage shows good performance with lifting mutation for DS1 with uniform crossover and Gaussian mutation compared to the data set without lifting allocation. From the set coverage result of E6.1 it was observed that the lifting allocation with P_r of 0.5 and

Algorithm 6.7: Lifting allocation mutation (c, P_r, σ_P, σ_A, λ_P, λ_A M, N)

```
M, N: number of houses, potential ITAP locations
c is a chromosome to be mutated
Lifting Allocation for houses
Probabilities rate P_r
if U([0,1]) < P_r then
    Lifting Allocation
    for i ∈ {1,...,M} do
        c[i] = max(min(c[i] + N(0,σ_A)1.0), 0)
    end
end
else
    for i ∈ {1,...,M} do
        if U([0,1]) < λ_A/M then
            c[i] = max(min(c[i] + N(0,σ_A)1.0), 0)
        end
    end
Mutating placement
for i ∈ {M+1,...,M+N} do
    Checking Mutation rate
    if U([0,1]) < λ_p/N then
        Gaussian mutation & clipping
        c[i] = max(c[i] + ⌊N(0,σ_P)⌋, 0)
    end
end
```

0.9 shows better performance than P_r (0.3) (see Table 6.14). For further investigation of DS1 the lifting allocation with P_r of (0.9) was then used.

Table 6.14: Set Coverage for Lifting Allocation in Uniform crossover and Gaussian mutation for DS1 of E6.1.

Set Coverage B	Set Coverage A					
	Lifting Allocation of Pr. 0.3		Lifting Allocation of Pr. 0.5		Lifting Allocation of Pr. 0.9	
	(B, A)	(A, B)	(B, A)	(A, B)	(B, A)	(A, B)
Without Lifting Allocation	7%	100%	7%	100%	8%	100%
Lifting Allocation prob. 0.3			64%	100%	75%	100%
Lifting Allocation prob. 0.5	100%	64%			93%	100%
Lifting Allocation prob. 0.9	100%	75%	100%	93%		

Applying lifting mutation to DS7 with arithmetic crossover and Gaussian mutation also showed an improvement. The result of E6.3 illustrates that lifting allocation with the Pr of 0.3, 0.5 and 0.9 outperformed the data set without lifting allocation, giving set coverage of 40%, 45% and 40%, respectively. The lifting allocation with Pr (0.5) outperformed the lifting allocation with Pr (0.3 and 0.9), as shown in Table 6.15. The potential of a good experimental result from the lifting allocation was perceived with Pr (0.5), presenting a good diversity of solutions in the search space (see Figure 6.10) which improved the performance of allocation with arithmetic crossover and Gaussian mutation.

In experiment E6.6 data set DS7 was investigated to demonstrate the set coverage for lifting allocation applied with arithmetic crossover and Gaussian mutation. The lifting allocation shows good performance compared to other cases. The lifting allocation for the Pr (0.5) did better than other rates (see Table 6.16). For further investigation of DS7 of E6.3 and E6.6 the lifting allocation with Pr (0.5) was used. The lifting allocation mutation showed a marked improvement in the diversity of the solutions in the search space of the sampled data sets.

Table 6.15: Set Coverage for Lifting Allocation in Arithmetic crossover and Gaussian mutation for DS7 of E6.3 .

Set Coverage B	Set Coverage A					
	Lifting Allocation of Pr. 0.3		Lifting Allocation of Pr. 0.5		Lifting Allocation of Pr. 0.9	
	(B, A)	(A, B)	(B, A)	(A, B)	(B, A)	(A, B)
Without Lifting Allocation	24%	40%	7%	45%	16%	40%
Lifting Allocation prob. 0.3			22%	28%	29%	31%
Lifting Allocation prob. 0.5	28%	22%			29%	26%
Lifting Allocation prob. 0.9	31%	29%	26%	29%		

Table 6.16: Set Coverage for Lifting Allocation in Arithmetic crossover and Gaussian mutation for DS7 of E6.6 .

Set Coverage B	Set Coverage A					
	Lifting Allocation of Pr. 0.3		Lifting Allocation of Pr. 0.5		Lifting Allocation of Pr. 0.9	
	(B, A)	(A, B)	(B, A)	(A, B)	(B, A)	(A, B)
Without Lifting Allocation	14%	42%	17%	54%	27%	33%
Lifting Allocation prob. 0.3			27%	32%	33%	36%
Lifting Allocation prob. 0.5	32%	27%			33%	23%
Lifting Allocation prob. 0.9	36%	33%	23%	33%		

6.3.5 Aggressive Placement Mutation

The previous section addressed the limitations of mutation for bandwidth allocation by applying a "lifting" adjustment to all houses. Here, modifying the mutation of placement genes by applying an "aggressive mutation" was proposed. Like lifting, ag-

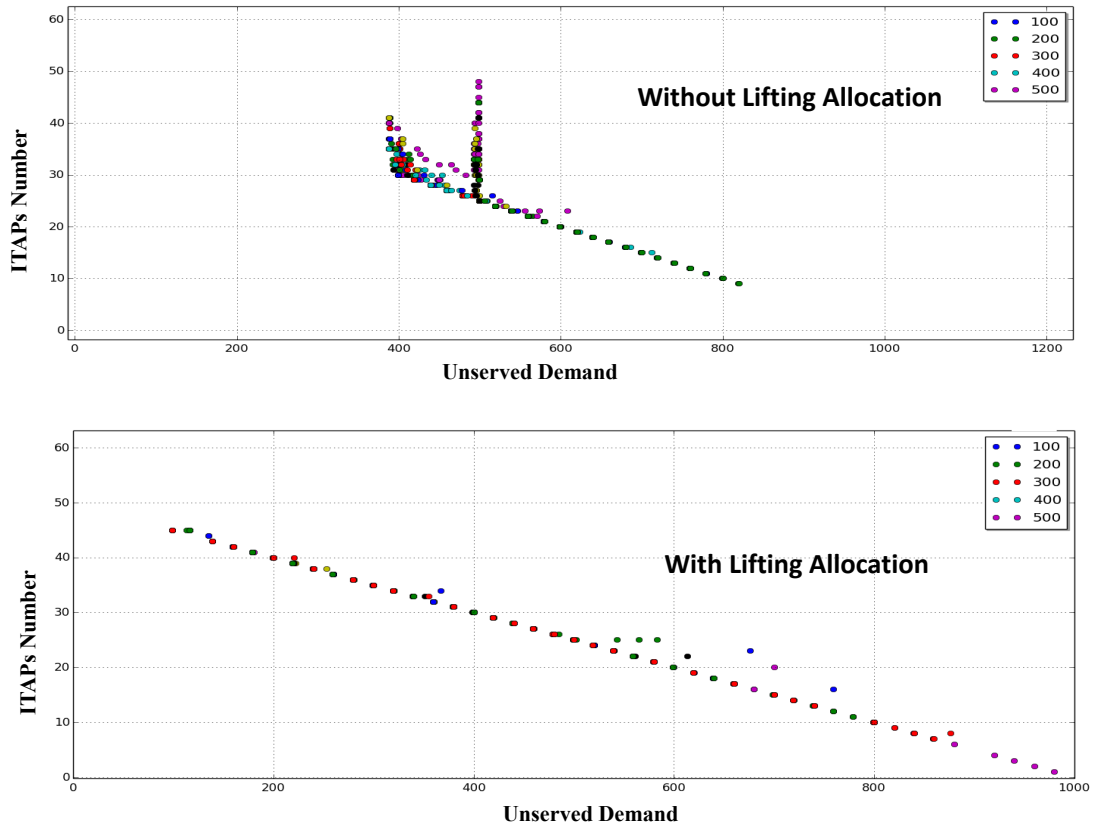

Figure 6.10: Diversity of solution in the search space without and with Lifting Allocation for DS7 of E6.3.

gressive identically mutates not a single gene but a whole chromosome with mutation probability λ_A/N, thereby making greater changes possible.

The selected mutating gene of the placement is swapped with the K value. The $K = 2, 5, 10$ value is a randomly chosen number from the set of 2, 5 and 10. The Algorithm 6.8 describes the approach.

The value of K used for aggressive placement mutation was compared in an experiment with Gaussian placement mutation, to see the effect of the changes that the aggressive mutation can make. In Table 6.17, the set coverage of aggressive placement mutation of value $K(= 2 \text{ and } 5)$ in DS1 of E6.1 showed a great improvement compared to the Gaussian placement mutation with lifting allocation. However, the aggressive placement mutation of value $K = 10$ showed no improvement over the

Algorithm 6.8: Aggressive placement mutation (c, P_r, K, σ_P, σ_A, λ_P, λ_A, M, N)

M, N: number of houses, potential ITAP locations
c is a chromosome to be mutated
Lifting Allocation, as above
Probabilities rate P_r
if $U([0,1]) < P_r$ **then**
    Lifting Allocation
    for $i \in \{1, ..., M\}$ **do**
        $c[i] = max(min(c[i] + N(0, \sigma_A)1.0), 0)$
    end
end
else
    for $i \in \{1, ..., M\}$ **do**
        if $U([0,1]) < \lambda_A/M$ **then**
            Gaussian mutation
            $c[i] = max(min(c[i] + N(0, \sigma_A)1.0), 0)$
        end
    end
Aggressive Mutation
for $i \in \{M+1, ..., M+N\}$ **do**
    Checking Mutation rate
    if $U([0,1]) < \lambda_p/N$ **then**
        Aggressive mutation
        $c[i] = U(\{0, 1, ..., K\})$
    end
end

Table 6.17: Set Coverage of Aggressive placement and lifting allocation over Gaussian mutation in E6.1, E6.3 and E6.6.

Set Coverage B Gaussian Mutation with Lifting Allocation	Set Coverage A Aggressive Placement Mutation with Lifting Allocation (0.9)					
	Aggressive mutation of K = 2		Aggressive mutation of K = 5		Aggressive mutation of K = 10	
	(B, A)	(A, B)	(B, A)	(A, B)	(B, A)	(A, B)
DS1 E6.1	91%	100%	91%	100%	90%	72%
DS7 E6.3	24%	30%	24%	30%	0%	19%
DS7 E6.6	18%	40%	15%	27%	7%	23%

Table 6.18: Best parameter values of lifting allocation and aggressive mutation

Data sets	Lifting Allocation Pr	Aggressive Mutation
DS1	0.9	2 and 5
DS7	0.5	2 and 5

Gaussian with lifting allocation. The set coverage of DS7 of E6.3 shows the aggressive placement mutation performance of value $K(= 2 \text{and} 5)$ compared to Gaussian mutation with lifting allocation that has the same improvement value. The aggressive placement mutation of value $K = 10$ shows a small improvement. The set coverage of DS7 of E6.6 demonstrates a greater improvement for aggressive placement mutation than Gaussian placement mutation with lifting allocation. The aggressive mutation of value K ensured a better performance than Gaussian placement mutation. Thus, the aggressive placement mutation of value K showed an improvement over the Gaussian placement mutation in all data sets, as shown in Table 6.17.

To summarise the lifting allocation and aggressive placement mutations performance with their parameters, for DS1 and DS7, see Table 6.18.

6.3.6 Mutation Rate

The mutation applied in sections 6.1 and 6.3 is controlled by a number of parameters. P_m controls whether an individual chromosome is mutated (otherwise, it is left unchanged), while P_r, λ_P, and λ_A, control the gene mutation in placement and allocation. Since the mutation rate can be very problem-specific, it is better to run experiments with several rates to see which rate maintains the greatest diversity in the population. Using too high a mutation rate will increases the diversity in the search space, but hinders convergence. At the same time, using too small a mutation rate may result in premature convergence (leading to local optima instead of a global optimum). In other words, too high a mutation rate reduces the search ability of NSGA-II to simple random sampling, while too small a mutation rate almost always fails, resulting in a local optimum due to the lack of diversity in the search space. In the previous experiments a mutation rate of 0.1 was used and then tests with mutation rates of 0.5 and 1.0, with lifting allocation were carried out, comparing these with the previous mutation rate. The set coverage values in Table 6.19 show that the mutation rate 0.5 outperformed the mutation rate of 0.1 in the data sets of experiments (E6.1, E6.3 and E6.6). The mutation rate 1.0 showed a slight improvement only in the large data sets of E6.3 and E6.6. The experimental results demonstrate that the set coverage for the mutation rate of 0.5 outperformed that of the mutation rate of 0.1, which indicates the effectiveness of the former over the latter. The best mutation rate seems to be in the range of 0.5 of population size 32.

6.3.7 Gene Probability Mutation

In earlier experiments gene mutation was applied with small probabilities of λ_P/N for placement and λ_A/M for bandwidth, where $\lambda_P = 1$ and $\lambda_A = 1$. Later, $\lambda_P = N$ and $\lambda_A = M$ were set so that every gene was mutated to see what effect this would have. The gene probability mutation of $\lambda_P = N$ and $\lambda_A = M$ plus mutation rates of 0.1

Table 6.19: Set Coverage of the mutation rate (0.5 and 1.0) for DS1, DS2 and DS3.

Set Coverage B **Mutation Rate**	**Set Coverage A** **Mutation Rate 0.1**					
	E6.1		**E6.3**		**E6.6**	
	(B, A)	**(A, B)**	**(B, A)**	**(A, B)**	**(B, A)**	**(A, B)**
Mutation Rate 0.5	100%	91%	30%	21%	37%	23%
Mutation Rate 1.0	82%	94%	33%	27%	30%	21%

Table 6.20: Gene mutation probability with mutation rate (0.1) for DS1, DS2 and DS3.

Set Coverage B	**Set Coverage A** **Gene Probability of $\lambda_P = N$ and $\lambda_A = M$**					
	E6.1		**E6.3**		**E6.6**	
	(B, A)	**(A, B)**	**(B, A)**	**(A, B)**	**(B, A)**	**(A, B)**
Gene probability of $\lambda_P = 1$ and $\lambda_A = 1$	100%	82%	30%	25%	43%	17%

and 0.5 were tested in the experiments (E6.1, E6.3 and E6.6). Generating high gene probability showed significantly worse results (no improvement) for the data sets than generating low gene probability with mutation rates of 0.1 and 0.5, as shown in Tables 6.20 and 6.21. The set coverage shows that generating a random gene probability of λ_P/N and λ_P/M gives a better result.

To sum up the experimental results obtained by the data sets with the suggested operators and parameters that were investigated with, see in Table 6.22 the conclusion of the genetic algorithm progress.

To test the algorithm and to evaluate the performance of the NSGA-II approach, the set coverage of NSGA-II and the weighted sum approach are compared for the data set samples of DS1, DS7 and DS7/E6.6, using the parameters of Table 6.22 plus Gaussian mutation. Table 6.23 shows that the NSGA-II algorithm outperforms the weighted sum

Table 6.21: Gene mutation probability with mutation rate (0.5) for E6.1, E6.3 and E6.6.

Set Coverage B	**Set Coverage A** Gene Probability of $\lambda_P = N$ **and** $\lambda_A = M$					
	E6.1		**E6.3**		**E6.6**	
	(B, A)	**(A, B)**	**(B, A)**	**(A, B)**	**(B, A)**	**(A, B)**
Gene probability of $\lambda_P = 1$ **and** $\lambda_A = 1$	100%	64%	32%	24%	42%	26%

Table 6.22: Summary of the used operators and parameters value with the Genetic Algorithm.

Data Set	**Crossover**	**Mutation**	**Lifting Allocation**	**Aggressive Mutation**	**Mutation Rate**	**Gene Probability**
DS1	Uniform	Gaussian	0.9	2 and 5	0.5	λ_P/N and λ_A/M
DS7	Arithmetic	Gaussian	0.5	2 and 5	0.5	λ_P/N and λ_A/M

approach for data sets DS1 and DS7. The non-dominated solutions of the weighted sum are clustered together while the solutions of NSGA-II are spread out in the search space; for example, in DS7 the maximum and minimum numbers of ITAPs in the weighted sum were 50 and 20, but in NSGA-II they were 49 and 0 respectively. This shows that the crowding distance in NSGA-II is higher and gives better results than the weighted sum.

Table 6.23: Set coverage of the NSGA-II and Weighted sum approaches

Set Coverage B	**Set Coverage A** NSGA II					
	DS1		**DS7**		**DS7 / E6.6**	
	(B, A)	**(A, B)**	**(B, A)**	**(A, B)**	**(B, A)**	**(A, B)**
Weighted Sum	9%	100%	0%	25%	0%	33%

6.4 Conclusion

In this chapter an evolutionary multi-objective optimization algorithm was presented to solve the problem of optimizing WMN infrastructure placement, defining the genetic algorithm which involves applying crossover and mutation operators on two individuals. The NSGA-II algorithm creates a child population from its parent population using fast non-dominated crossover and mutation. Several initial solutions with different operators and parameters were investigated, to ensure the presence of diversity, aiming to identify the best operator and parameters to use as part of the comprehensive solution methodology. The literature mentions a great variety of crossovers; the ones illustrated here were *Arithmetic*, *Uniform* and *Hybrid* crossovers. The set coverage comparison of crossover indicated that the uniform crossover outperformed the arithmetic and hybrid crossovers in small data sets such as DS1 and DS1A, while the arithmetic crossover outperformed the uniform and hybrid crossovers in big data sets such as DS7 and DS7A. The experimental results on the data sets samples indicate that the Gaussian mutation outperformed uniform mutation and was effective as a genetic operator. However, the results for arithmetic crossover and Gaussian mutation dramatically improved when used in combination with a lifting allocation of 0.5 probability. Further investigation of the potential of these operators is suggested. To change the mutation of the placement genes an aggressive placement mutation was applied; the results of this experiment demonstrated the performance of aggressive placement mutation compared to Gaussian placement mutation. A random mutation was applied to one or more genes for the earlier test of instances the mutation rate was set to 0.1 and then mutation rates of 0.5 and 1.0 were applied. The experimental result of the mutation rate at 0.5 outperformed the mutation rate at 0.1 and 1.0. The gene probability mutations of $\lambda_P = N$, $\lambda_A = M$, λ_P/N and λ_A/M were applied and it could then be observed that the high gene probability mutation showed no improvement in the data set samples. The greatest performance improvements of mutation were obtained by using a mutation rate of 0.5 and a gene probability mutation of λ_P/N and λ_A/M. To

conclude this chapter, the NSGA-II algorithm results were compared with the results of using the weighted sum approach. The set coverage result showed that the NSGA-II outperformed the weighted sum approach for the sampled data sets. This indicates the effectiveness and good performance of NSGA-II. It was observed from the experiments that the NSGA-II algorithm performed generally better than the weighted sum approach in terms of diversity and quality from the approximation of the Pareto front.

Chapter 7

Conclusion and Suggestions for Future Work

The conclusion of this thesis provides a summary of the research into problems of WMN infrastructure placement problems and explores possible future directions.

7.1 Thesis Summary and Contributions

The thesis examined the following hypothesis:

- With single objective approach using meta-heuristic algorithms such as Hill Climbing and Simulated Annealing algorithms, can effectively design WMNs by using a suitable combination of move operators to place the ITAPs to improve performance and efficiency, this technique provides a better service such as minimising the number of ITAPs and maximising throughput in a WMN.
- With suitable crossover and mutation operators, a multi-objective genetic algorithm approach is more effective than a weighted sum approach in producing a range of networks that encapsulate the trade-off between the number of ITAPs, throughput and fairness of bandwidth allocation in a WMN.

The aim was to investigate combinations of move operators that could be used in heuristic and meta-heuristic approaches to efficiently solve such placement problems ex-

pressed as single and multi-objective formulations.

After reviewing the related literature, our initial study considered two complementary single objective formulations. In Chapter 3, using an ideal link model, we demonstrated the effectiveness of simple move operators for HC and SA algorithms in maximize the throughput of WMNs with a fixed number of ITAPs. Experimental evaluation showed the efficiency of combining the "Swap" and "Reallocate" move operators and the superiority of the SA algorithm in achieving better solutions than the HC algorithm.

Chapter 4 examined the converse formulation, namely, minimizing the number of ITAPs that need to be installed to satisfy all consumer demand, by adding "Delete" and "Add" move operators to "Swap" and "Reallocate". The experimental evaluation shows that the combination of all four move operators is efficient because it provides a good solution for the placement of ITAPs in WMNs. Demonstrating the maximizing of throughput and minimizing of the number of ITAPs, the parameters used in the model revealed the great impact on the network performance of optimizing the ITAP placement.

Chapter 5 introduced a multiple objective formulation of the problem and applied the weighted sum method to HC and SA, demonstrating the significance of the choice of weights for the ability of the algorithms to explore the whole search space. The result indicated a strong effect from a number of different starting conditions (e.g. initial bandwidth allocation), resulting in a clear bias towards individual objectives. More aggressive neighbourhood move operators were proposed to enable the optimization algorithms to make larger jumps across the search space and converge more rapidly. This approach showed the ability to balance the three objectives under review and a range of solutions was explored to cover the search space.

In Chapter 6 , an evolutionary multi-objective optimization algorithm was presented in order to solve the problem of optimizing WMN infrastructure placement, defining the crossover and mutation operators based on the concepts of the neighbourhood moves in Chapter 5. It was observed from the experiments that the evaluation results

of the NSGA-II algorithm on the data sets sampled achieved better solutions than the weighted sum approach in terms of diversity and quality from the approximation of the Pareto front in optimizing solutions to the problem of WMN infrastructure placement.

7.2 Future Work

This thesis has made a step towards the automated planning of WMN that can be implemented to provide cost effective wireless coverage for a huge area, focusing on the placement of infrastructure.

For the success of future WMNs, if services with such QoS requirements as bandwidth, delay and reliability are to be developed, a fundamental issue must be taken care of. The energy consumption must be taken into account in any future work regarding cost issues. Wireless signals operating at similar frequencies can interfere with each other, with significantly negative effect on the performance of the network. The performance problems occur for many reasons, such as collision, multi-path interference and traffic slow-downs due to congestion. Therefore, applying realistic models such as a general model or real world model to regulate interference in WMNs is desirable.

Having demonstrated the potential to optimize the trade-off between multiple objectives, further work could be devoted to expanding the range of measures and investigating the interplay between them. For example, little attention has hitherto been given to energy efficiency as part of an optimized framework, but this has clear future implications, particularly considering that consumers must use their energy consumption to provide service to others. Considering aspects of routing and scheduling also has the potential to improve the solutions overall. In the future, more complex models of the network could be used, using, for example, hybrid routing mechanisms with multicast destinations, QoS based routing, bandwidth reservation, delay constraints and packet loss to a multicast session for some region. The idea of this future direction is to design routing schemes which adopt bandwidth estimation techniques; this could

which increase the data delivery rate with less network bandwidth.

Bibliography

[1] H. of Parliament, "Uk broadband infrastructure," 2015. Available at http://researchbriefings.files.parliament.uk/documents/POST-PN-0494/POST-PN-0494.pdf [Online; accessed 2017-03-28].

[2] R. Chandra, L. Qiu, K. Jain, and M. Mahdian, "Optimizing the placement of internet taps in wireless neighborhood networks," in *Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference on*, pp. 271–282, IEEE, 2004.

[3] B. Aoun, R. Boutaba, Y. Iraqi, and G. Kenward, "Gateway placement optimization in wireless mesh networks with qos constraints," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 2127–2136, 2006.

[4] J. Wang, B. Xie, K. Cai, and D. P. Agrawal, "Efficient mesh router placement in wireless mesh networks," in *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE International Conference on*, pp. 1–9, IEEE, 2007.

[5] M. L. Sanni, A.-H. A. Hashim, F. Anwar, A. W. Naji, and G. S. Ahmed, "Gateway placement optimisation problem for mobile multicast design in wireless mesh networks," in *Computer and Communication Engineering (ICCCE), 2012 International Conference on*, pp. 446–451, IEEE, 2012.

[6] F. Xhafa, C. Sanchez, L. Barolli, and R. Miho, "An annealing approach to router nodes placement problem in wireless mesh networks," in *Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on*, pp. 245–252, IEEE, 2010.

[7] E. Amaldi, A. Capone, M. Cesana, I. Filippini, and F. Malucelli, "Optimization models and methods for planning wireless mesh networks," *Computer Networks*, vol. 52, no. 11, pp. 2159–2171, 2008.

[8] IBM, "CPLEX optimizer," 2012. Available at `https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/` [Online; accessed 2013-05-30].

[9] S. M. Allen, I. M. Cooper, and R. M. Whitaker, "Optimising multi-rate link scheduling for wireless mesh networks," *Computer Communications*, vol. 35, no. 16, pp. 2014–2024, 2012.

[10] A. Capone, I. Filippini, and F. Martignon, "Joint routing and scheduling optimization in wireless mesh networks with directional antennas," in *Communications, 2008. ICC'08. IEEE International Conference on*, pp. 2951–2957, IEEE, 2008.

[11] D. Benyamina, A. Hafid, and M. Gendreau, "Wireless mesh networks design - a survey," *IEEE Communications surveys & tutorials*, vol. 14, no. 2, pp. 299–310, 2012.

[12] A. Roy and G. Umapathi, "Study on resource allocation in wireless mesh network," *International Journal of Emerging Technology and Advanced Engineering*, vol. 4, no. 1, pp. 502–506, 2014.

[13] F. Xhafa, C. Sanchez, and L. Barolli, "Ad hoc and neighborhood search methods for placement of mesh routers in wireless mesh networks," in *Distributed Computing Systems Workshops, 2009. ICDCS Workshops' 09. 29th IEEE International Conference on*, pp. 400–405, IEEE, 2009.

[14] K. Fleszar, I. H. Osman, and K. S. Hindi, "A variable neighbourhood search algorithm for the open vehicle routing problem," *European Journal of Operational Research*, vol. 195, no. 3, pp. 803–809, 2009.

[15] F. Li, Y. Wang, X.-Y. Li, A. Nusairat, and Y. Wu, "Gateway placement for throughput optimization in wireless mesh networks," *Mobile Networks and Applications*, vol. 13, no. 1-2, pp. 198–211, 2008.

[16] J. Jun and M. L. Sichitiu, "The nominal capacity of wireless mesh networks," *IEEE wireless communications*, vol. 10, no. 5, pp. 8–14, 2003.

[17] J. Tang, G. Xue, and W. Zhang, "Maximum throughput and fair bandwidth allocation in multi-channel wireless mesh networks.," in *INFOCOM*, vol. 00, pp. 1–10, 2006.

[18] S. Nahle and N. Malouch, "Fairness enhancement in wireless mesh networks," in *Proceedings of the 2007 ACM CoNEXT conference*, p. 30, ACM, 2007.

[19] J. Hoblos, "Fairness enhancement using transmission scheduling in multi-hop wireless mesh networks," in *Communications and Information Technology (ICCIT), 2013 Third International Conference on*, pp. 251–255, IEEE, 2013.

[20] O. Gurewitz, V. Mancuso, J. Shi, and E. W. Knightly, "Measurement and modeling of the origins of starvation of congestion-controlled flows in wireless mesh networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 6, pp. 1832–1845, 2009.

[21] J. Hoblos and H. Peyravi, "Fair access rate (far) provisioning in multi-hop multi-channel wireless mesh networks," in *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2010 International Congress on*, pp. 68–73, IEEE, 2010.

[22] T. Nandagopal, T.-E. Kim, X. Gao, and V. Bharghavan, "Achieving mac layer fairness in wireless packet networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 87–98, ACM, 2000.

[23] S. Allen and R. Whitaker, "The effect of consumer demand on optimised wireless mesh network deployment," in *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*, pp. 906–910, IEEE, 2009.

[24] S. M. Allen, R. M. Whitaker, and S. Hurley, "Personalised subscription pricing for optimised wireless mesh network deployment," *Computer Networks*, vol. 52, no. 11, pp. 2172–2188, 2008.

[25] T. Szymanski, "Achieving minimum-routing-cost maximum-flows in infrastructure wireless mesh networks," in *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pp. 2031–2036, IEEE, 2012.

[26] T. H. Szymanski, "Max-flow min-cost routing in a future-internet with improved qos guarantees," *IEEE Transactions on Communications*, vol. 61, no. 4, pp. 1485–1497, 2013.

[27] J. Robinson, M. Uysal, R. Swaminathan, and E. Knightly, "Adding capacity points to a wireless mesh network using local search," in *INFOCOM 2008. The*

27th Conference on Computer Communications. IEEE, pp. 1247–1255, IEEE, 2008.

[28] S. A. Curtis, "The classification of greedy algorithms," *Science of Computer Programming*, vol. 49, no. 1-3, pp. 125–157, 2003.

[29] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[30] H. Holland John, "Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence," *USA: University of Michigan*, 1975.

[31] G. R. Raidl, "An improved genetic algorithm for the multiconstrained 0-1 knapsack problem," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pp. 207–211, IEEE, 1998.

[32] L. M. Schmitt, "Theory of genetic algorithms ii: models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary fitness function under scaling," *Theoretical Computer Science*, vol. 310, no. 1-3, pp. 181–231, 2004.

[33] M. N. H. Siddique and M. O. Tokhi, "Ga-based neuro-fuzzy controller for flexible-link manipulator," in *Control Applications, 2002. Proceedings of the 2002 International Conference on*, vol. 1, pp. 471–476, IEEE, 2002.

[34] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: An autocatalytic optimizing process," *TR91-016, Politecnico di Milano*, pp. 1–21, 1991.

[35] M. Dorigo, G. Di Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial life*, vol. 5, no. 2, pp. 137–172, 1999.

[36] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[37] V. T. Le, N. H. Dinh, and N. G. Nguyen, "A novel pso-based algorithm for gateway placement in wireless mesh networks," in *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pp. 41–45, IEEE, 2011.

[38] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of machine learning*, pp. 760–766, Springer, 2011.

[39] H. D.-N. Le, N. G. Nguyen, N. H. Dinh, N. D. Le, and V. T. Le, "Optimizing gateway placement in wireless mesh networks based on aco algorithm," *International Journal of Computer and Communication Engineering*, vol. 2, no. 2, p. 143, 2013.

[40] D.-N. Le, N. G. Nguyen, N. D. Le, and V. T. Le, "A new evolutionary approach for the optimal location of controllers in wireless networks," in *The 2nd International Conference on Information Communication and Management (ICICM 2012)*, 2012.

[41] H. Leung and D. D. Lu, *A PSO Approach in Optimal FACTS Selection with Harmonic Distortion Considerations*. INTECH Open Access Publisher, 2013.

[42] P. Shelokar, P. Siarry, V. K. Jayaraman, and B. D. Kulkarni, "Particle swarm and ant colony algorithms hybridized for improved continuous optimization," *Applied mathematics and computation*, vol. 188, no. 1, pp. 129–142, 2007.

[43] K. Deb, "Multi-objective optimisation using evolutionary algorithms: An introduction," *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, pp. 3–34, 2011.

[44] K. Deb and T. Goel, "Evolutionary multi-criterion optimization," *Evolutionary algorithms in engineering and computer science*, pp. 135–162, 1999.

[45] D. Benyamina, A. Hafid, and M. Gendreau, "Wireless mesh network planning: A multi-objective optimization approach," in *Broadband Communications, Networks and Systems, 2008. BROADNETS 2008. 5th International Conference on*, pp. 602–609, IEEE, 2008.

[46] L. Zadeh, "Optimality and non-scalar-valued performance criteria," *IEEE transactions on Automatic Control*, vol. 8, no. 1, pp. 59–60, 1963.

[47] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, vol. 16. John Wiley & Sons, 5 2001.

[48] K. Proos, G. Steven, O. Querin, and Y. Xie, "Multicriterion evolutionary structural optimization using the weighting and the global criterion methods," *AIAA journal*, vol. 39, no. 10, pp. 2006–2012, 2001.

[49] S. Saramago and V. Steffen, "Optimization of the trajectory planning of robot manipulators taking into account the dynamics of the system," *Mechanism and machine theory*, vol. 33, no. 7, pp. 883–894, 1998.

[50] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[51] O. Rifki and H. Ono, "A survey of computational approaches to portfolio optimization by genetic algorithms," in *18th International Conference Computing in Economics and Finance*, 2012.

[52] X. Wang, S.-h. Yu, T. Luo, and J. Dai, "An improved multiple objectives optimization of qos routing algorithm base on genetic algorithm," in *Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on*, pp. 1–4, IEEE, 2009.

[53] M. Camelo, C. Omana, and H. Castro, "Qos routing algorithms based on multiobjective optimization for mesh networks," *IEEE Latin America Transactions*, vol. 9, no. 5, pp. 875–881, 2011.

[54] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

[55] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[56] T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.

[57] L. Nawaf, C. Mumford, and S. Allen, "Optimizing the placement of itaps in wireless mesh networks by implementing hc and sa algorithms," in *International Conference on Ad Hoc Networks*, pp. 29–41, Springer, 2015.

[58] T. Okabe, Y. Jin, and B. Sendhoff, "A critical survey of performance indices for multi-objective optimisation," in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 2, pp. 878–885, IEEE, 2003.

[59] L. Nawaf, S. M. Allen, and O. Rana, "Internet transit access point placement and bandwidth allocation in wireless mesh networks," in *Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual*, pp. 1–8, IEEE, 2017.

[60] G. S. Ladkany and M. B. Trabia, "A genetic algorithm with weighted average normally-distributed arithmetic crossover and twinkling," *Applied Mathematics*, vol. 3, no. 10A, pp. 1220–1235, 2012.

[61] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, (San Francisco, CA, USA), pp. 2–9, Morgan Kaufmann Publishers Inc., 1989.

[62] R. Tinós and S. Yang, "Evolutionary programming with q-gaussian mutation for dynamic optimization problems," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pp. 1823–1830, IEEE, 2008.

[63] A. V. Kononova, D. W. Corne, P. De Wilde, V. Shneer, and F. Caraffini, "Structural bias in population-based algorithms," *Information Sciences*, vol. 298, pp. 468–490, 2015.