HEURISTIC ALGORITHMS FOR WIRELESS MESH NETWORK PLANNING

SHAH MOSTAFA KHALED
Master of Science, University of Dhaka, 2004

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

HEURISTIC ALGORITHMS FOR WIRELESS MESH NETWORK PLANNING

SHAH MOSTAFA KHALED

Approved:

Signature *Date*

Supervisor: Dr. Robert Benkoczi

Committee Member: Dr. Shahadat Hossain

Committee Member: Dr. Mark Walton

Chair, Thesis Examination Committee: Dr. Howard Cheng

To *Fariha Sharmin*, my lovely wife
who has been eagerly waiting for this accomplishment since a while

Abstract

Technologies like IEEE 802.16j wireless mesh networks are drawing increasing attention of the research community. Mesh networks are economically viable and may extend services such as Internet to remote locations. This thesis takes interest into a planning problem in IEEE 802.16j networks, where we need to establish minimum cost relay and base stations to cover the bandwidth demand of wireless clients. A special feature of this planning problem is that any node in this network can send data to at most one node towards the next hop, thus traffic flow is unsplittable from source to destination.

We study different integer programming formulations of the problem. We propose four types of heuristic algorithms that uses greedy, local search, variable neighborhood search and Lagrangian relaxation based approaches for the problem. We evaluate the algorithms on database of network instances of 500-5000 nodes, some of which are randomly generated network instances, while other network instances are generated over geometric distribution. Our experiments show that the proposed algorithms produce satisfactory result compared to benchmarks produced by generalized optimization problem solver software.

Acknowledgments

First and foremost, I would like to thank Dr. Robert Benkoczi for his support and guidance. This thesis could not have been written without him. He not only served as my supervisor but also encouraged and challenged me throughout my M.Sc. program accepting nothing less than my best efforts. Many thanks to my committee members Dr. Shahadat Hossain and Dr. Mark Walton for helping me in directing my work and for reading my thesis and giving me feedbacks.

I would like to thank all my fellow graduate students Mr. Stephen Nsoh, Mr. Ben Burnett, Mr. Ahmed Tahsin Zulkarnine especially Mr. Khobaib Zaamout for the help he provided to me throughout my stay in this country. I also would like to thank my wife Fariha Sharmin for her support on the home front, cooking, cleaning, and attending to me without a complaint. For that I will always be in her debt.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Access to information has become very important in the modern life. The Internet, the staple source of information today, is growing richer and larger everyday. The demand on network bandwidth is also growing as the services and contents on the Internet become increasingly data intensive. In this context, broadband access to the Internet has become an important issue.

The success of cellular telecommunication industry has led the business and research community to further consider the importance of wireless access to the Internet. Many devices like laptops, personal digital assistants, etc. are widely used, and require wireless access to the Internet. To address this demand, the wireless industry has also been delivering technologies such as WiFi, WiMAX, GSM 3G, etc. Historically, wireless subscribers connected to base stations which were attached to wired networks. Later, when the demand for coverage grew, network relaying was introduced extend coverage. The benefit of this was two fold: first, it facilitated the expansion of networks, and second, it gave a wireless alternative to the cable access networks for last mile connectivity. Future wireless systems are expected to meet higher demands on the quality of service in terms of data rate, reliability, and cost, among others.

The research in this thesis deals with the design of wireless networks and relaying. It essentially intends to optimize the establishment cost of a wireless network given information about demand, capacity, cost of subscribers, relays and base stations.

In the following section we discuss the development of wireless technologies, the driving forces behind the development, and some open questions. We focus particularly on IEEE 802.16j family of wireless mesh networks, which is the underlying technology considered in this thesis.

1.1 Background

Wireless mesh network [4] is a communication technology where wireless nodes communicate with each other in a mesh topology. In a mesh network communication, devices are generally categorized into clients, routers, and gateways. In such networks all gateways may directly provide a service (for example Internet) and routers relay the data that must flow between subscribers and gateways. Wireless mesh networking has been used in technologies such as IEEE 802.11, IEEE 802.15, IEEE 802.16, etc [130]. assumed as the carrier provider for the design problem Following the great success of WiFi (IEEE 802.11) in replacing old wired computer networks from households and offices, the focus of the industry and research has been towards eliminating the cost to setup and maintain the cabling for broadband metropolitan area network. This led to the development of IEEE 802.16, which is now serving as the backhaul[1] of broadband wireless access for wireless metropolitan area network (WMAN) [9]. The inter-operable implementations of the IEEE 802.16 family of wireless network standards is popularly known as WiMAX [9]. The IEEE Standards Board established a working group in 1999 to develop standards for broadband WMAN. In February 2004 this group published the first draft for IEEE 802.16. In this stage the subscriber stations (SS) were supposed to be immobile and in line-of-sight with the base stations (BS) [134].

The next significant development was the introduction of IEEE 802.16e-2005, dealing with mobility of the subscribers and non-line-of-sight communication between SS and BS, better support for quality of service, use of Scalable Orthogonal Frequency Division Multiple Access, etc [134]. Even with this development practical problems existed, such as signal-to-noise-ratio at the cell edge, coverage holes that existed due to shadowing and

[1] In a hierarchical telecommunications network the backhaul portion of the network comprises the intermediate links between the core network, or backbone network and the small subnetworks at the edge of the entire hierarchical network (Wikipedia).

non-light-of-sight connections, the access requirement of non-uniform distributed traffic in densely populated areas, etc. In competition with 3G and wired broadband providers, the WiMAX protocols had to ensure more reliability, had to fill coverage holes, had to support for maximum mobility. These challenges are conflicting to each other. Increasing data rate reduces reliability. On the other hand, increase in reliability of service would reduce coverage area (i.e. the cell size). Reduction of cell size would result in necessity of higher number of BSs to cover a given area, which would increase the cost of the network [134]. The most promising solution at that point was to insert some low cost relay stations (RS) between the SSs and BSs, which will route data in between. These relays are used to extend the capacity and coverage area of the network, and bridge the coverage holes such as areas in the shadows of buildings and thereby enhance the quality of end-to-end communication [75]. IEEE 802.16j was an amendment to IEEE 802.16e. It proposed the mobile multi-hop relay (MMR) network, which takes advantage of the multi-hop wireless connectivity where data between a BS and SS can be relayed through a RS [75].

IEEE 802.16j widened the scope of WiMAX in terms of increasing throughput and coverage, giving a solution to the coverage holes inside buildings, and it made it easy to extend temporary coverage to densely populated areas. The network architecture introduces many complexities within the already challenging environment of radio access networks with mobility support [75], for example channel access scheduling in terms of frequency and time, frequency reuse, RS and BS placement, resource (frequency, time) allocation, etc [134]. In the following paragraphs we give examples of different types of research conducted in IEEE 802.16j networks to get a high level idea about them and then survey research on wireless network design problems which are similar to the problem considered in this thesis.

In wireless communication the stations in the network share the same communication medium and interference may occur. To resolve this situation, network resources such as

time and frequency were split in time slots and frequency channels. Data transmission is assigned a slot in time as well as a frequency channel so that nodes that interfere with each other receive different time slots and channels. Given some network stations in an area, and resources like time slots and frequency channels, a question is how to allocate these resources to the stations so that the stations can transmit data without interfering other stations, and ensuring the best use of these resources. Medium Access Control (MAC) provides access control mechanisms to wireless medium that make it possible for several stations to communicate within a multiple access network that incorporates a shared medium. A *frame* is a digital data transmission unit that includes a sequence of bits making it possible for the receiver to detect the beginning and end of the data unit in the stream of bits called frame synchronization sequence. If a receiver is connected to the system in the middle of a frame transmission, it ignores the data until it detects a new frame synchronization sequence. Zhang et. al. studied the performance of a MAC protocol for IEEE 802.16j networks [164]. Tao et. al. in [150] proposed a new frame structure to optimize relay-based transmissions in IEEE 802.16e networks. Methods to reduce the overhead of traffic relaying in the network were proposed in [109]. Erwu et. al. in [56] proposed a down link bandwidth allocation algorithm designed to dynamically allocate resources on the down link for RSs transmission based on the per frame bandwidth requirement. Tao et. al. [151] proposed an aggregation scheme for traffic on the relay links (i.e., BS-RS or RS-RS links) to reduce the system overhead and thus improve the efficiency of utilization of the MAC frame. Kwon et. al. proposed a scheduling scheme for the relay stations in IEEE 802.16j MMR networks in [108]. To enable MMR networks satisfy traffic demand from different user groups Erwu et. al. proposed a bandwidth allocation scheme in [56]. In the context of applying spatial reuse, an analytical model was used to investigate the system capacity with varying number of RSs and their transmission power [74]. Sundaresan et. al.

in [149] proposed scheduling algorithms to help spatial reuse gains from the two hops[2] of the relay-enabled network. Paschos et. al. in [131] showed that spatial reuse could mitigate the capacity loss, while they were studying an analytical model to investigate cell capacity with two-hop coverage extension. Other than these there has been a substantial amount of research focusing on the locational design of wireless networks.

Architectural and topological planning for networks have been a well investigated area for different wireless networks. Amaldi et. al. used an integer programming model, and a number of algorithms based on greedy and tabu search [78] to determine better locations of BSs to cover different traffic densities [8]. A big problem for the cellular phone industry was to migrate from 2G to 3G networks in such a way that minimum number of cells are used with minimized associated cost to satisfy the user demands. Abusch-Magder in [2] proposed a heuristic[3] algorithm to determine cell sites, the heuristic worked on ranking cells based on data generated in simulation and periodic removal of cells from the simulation model.

Initial studies on the design of IEEE 802.16j system have been carried out by Lin et. al. [119]. The work presented in [118] formulates an integer programming model on IEEE 802.16d networks and proposes heuristic algorithms to solve the problem of designing a minimum cost backhaul wireless network (with BSs) to satisfy SS demands without violating capacity constraints at the BSs. A numerical case study has been presented in the following work by Lin et. al. in [116] to find the optimal location of a relay to maximize the system throughput within the RS-BS cell coverage in 802.16j networks. The work, however, does not address the issues related to the planning of multiple relays. Fu et. al.

[2]A hop is an intermediate link in a sequence of links connecting two network devices. On the Internet, data packets go through several routers before they reach destination. Each time the packet is forwarded to the next router is called a hop.

[3]A heuristic is a computational technique for problem solving that does not guarantee optimality of the solution. It is a simple algorithm based on some idea or observation applied on a difficult problem that is thought to produce an acceptable solution.

[69] studied the problem of network deployment and radio resource reuse in IEEE 802.16j MMR networks and a heuristic-based placement of RSs was introduced. The capacity of an IEEE 802.16j network for up link transmissions using cooperative diversity was evaluated in [157]. The outcome of this research can be used for analyzing the trade-off between relay deployment cost and capacity improvement.

Yu et. al. in [163] described an integer programming formulation for placement of RS and BS with an objective to minimizing the cost of their establishment under the constraint on traffic demand from the users. Standard branch and bound techniques were used to solve the problem. Although this approach could solve problems of small instances, it could not handle large instances of metropolitan scale. A clustering approach was applied to the same problem in [162]. A heuristic algorithm was proposed by Lin et. al. [117] to obtain the minimum number of RSs in an MMR network.

In 2009 Niyato et. al. [129] presented a relay centric hierarchal optimization model for jointly optimizing the radio resource management and network planning for the RSs in MMR networks. The objectives were to maximize the utilization of the RSs, and obtain the optimal amount of reserved bandwidth. In another algorithm they formulated optimization problem based on chance-constrained assignment problem to obtain the optimal decisions on relay placement and base station selection.

In 2010 Ge et. al. [73] published a paper on using IEEE 802.16j technology to improve vehicle to infrastructure communication in vehicular wireless network. They assumed the locations of vehicular SSs known, and based on that information tried to obtain the optimal placement of RSs such that the end to end capacity is maximized. This study incorporated a highway mobility model and formed the problem as a nonlinear optimization model. The solution of such a model guarantees maximal end to end capacities to SSs.

1.2 Research Problem

The problem considered in this research is a special case of IEEE 802.16j network planning, where we have at most one layer of relays between the SSs and BSs. It is also assumed that a SS can get service from at most one RS or BS, and a RS can route data through at most one BS. This is to make sure that all data packets routed between SSs and BSs in this network are sequenced in order they were generated, thus simplifying the overall design of the network protocol. Throughout the rest of the thesis we call this property 'unsplittable flow' property of the problem. The objective of the problem is to determine the optimal placement of BSs and RSs given a set of candidate sites, demand of the SSs, cost and capacity information of the BSs and RSs. Such placement would also guarantee the optimal association between BS-RS, RS-SS and BS-SS to ensure quality of service parameters such as speed, coverage, throughput, etc.

Why this design problem is computationally hard and why it is interesting will be discussed in detail in later parts of this thesis. We represent the problem in integer programming formulations. We propose local search based heuristic algorithms to get a good approximated solution in short computational time. We also propose Lagrangian relaxation based approaches on the problem.

1.3 Organization of the Thesis

Although there is a good volume of literature addressing WiMAX wireless network design problems both from industry and research, we could not find any literature specifically addressing the problem studied in this thesis. However, we discuss different optimization problems which are related to our research problem such as Set Cover, Capacitated Set Cover, Bin Packing and Unsplittable Maximum Flow in Chapter 2. We also review the

available literature on these problems to get an idea about how researchers have addressed these NP-hard problems.

In Chapter 3 we present two integer programming models of the network design problem. We also introduce local search and a variable neighborhood search based heuristic algorithm for minimum cost network design.

Chapter 4 presents another integer programming formulation of the network design problem and a Lagrangian relaxation based heuristic algorithm for minimum cost network design. This heuristic also gives us a lower bound to the solution.

Chapter 5 presents the experimental results based on the algorithm described in Chapter 3 and 4.

Chapter 6 concludes the thesis with future research directions.

Chapter 2

Related Research

In previous chapter we have reviewed different problems in wireless mesh network and IEEE 802.16j in particular. We observed that IEEE 802.16j introduced different optimization problems relating to placement of network stations, scheduling and allocation of time and frequency channel, etc. Majority of these problems are intended to maximize use of bandwidth capacity, minimize cost, reduce relaying overhead, maximize system throughput, maximize reuse of radio resources, etc. The research under this thesis deals with a wireless network planning problem of optimal cost placement of wireless relays and base stations under certain constraints of capacity and flow. From the definition of the problem presented in section 2.1, we see that the problem is characterized by location-allocation, capacity, flow conservation and cost minimization properties. Later in sections 3.1.1, 3.1.2 and 4.1 we have formulated three mathematical models to represent the problem. It is also evident from the model in section 4.1 that our research problem can be modeled in terms of packing and covering constraints. This model uses some trees that combinatorially maintain the properties of capacity constraint, flow conservation and unsplittable flow, which are explicitly used to formulate models in sections 3.1.1 and 3.1.2. We could not find any literature exactly addressing our research problem, and therefore we review problems that contain different aspects of our research problem with the intention of finding guidance about approaches to solve our research problem.

We compare our research problem to three classic computationally hard problems Set Cover, Bin Packing and Capacitated Set Cover and establish that our research problem generalizes all these three problems, and thereby prove its computational hardness. These problems have the notion of location-allocation, capacity, cost and flow, that most network research problems contain. It is interesting to study these problems to see how different al-

gorithms applied to these problems can be useful to solve network design problems. We review the research conducted on these three problems to see what solution approaches have been applied to handle these problems and measure the quality of solution those achieved. Since our research problem, by its definition, also relates to unsplittable flow problems, we also review the research literature available in this area. In the rest of this thesis we will refer to our research problem as 'WiMAX planning problem'.

2.1 Problem Statement

The WiMAX planning problem has a set of wireless subscriber stations (SS) with their bandwidth demands, and some candidate sites where we can establish wireless relay stations (RS) and base stations (BS) to cover the demands of SSs. The cost of establishment and bandwidth capacities for these RS and BS are also given. In addition, the channel gains [1] between the locations of candidate sites and SSs, and between RS candidate sites and BS candidate sites can also be obtained. We assume that there is no power control mechanism at either end of the channel. From these channel information we can find which RSs and BSs are close to a SS to connect with, and which BSs are around a RS to connect. The problem we need to solve in this network is to minimize the total cost of establishing RSs and BSs to cover the demands of all the SSs. Another constraint intended to enhance the performance of the network in design is that traffic from any SS or RS in this network cannot be split when sent to the next hop. This means that all the demand of a SS has to be served by no more than one RS or BS. Similarly all the traffic going through a RS has to be served by no more than one BS. Thus we are supposed to choose a subset of RS and BS, and assign traffic to no more than one link for each SS, and do the same for all the open RSs while connecting to BSs. BSs and RSs are nodes in the wireless network responsible

[1] The ability of a receiver station to recover the signal that was sent by its peer.

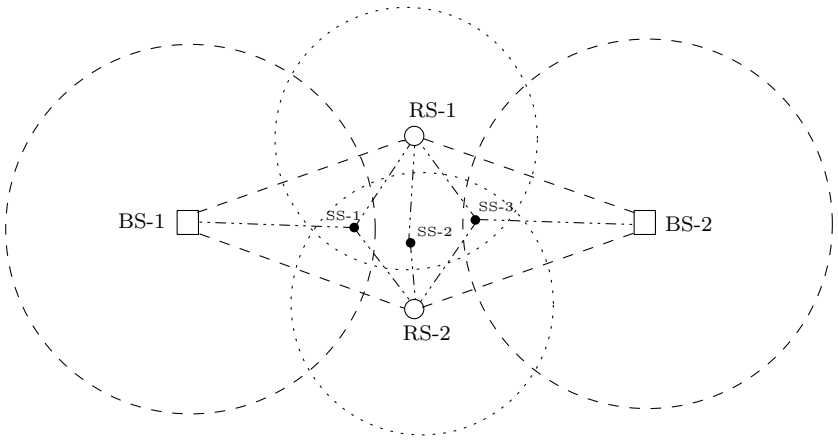


Figure 2.1: Example problem instance

for routing data. Generally BSs have greater capacity, coverage area and installation cost than that of RSs. In the context of this problem, we assume that BSs are connected to a high speed wired network, as a result a SS must connect to a BS to get Internet service. RS have extended the coverage area of the network.

The problem has two aspects: first is the optimal cost selection of candidate sites for RSs and BSs such that they have enough capacity to meet the demand of the SSs. The second aspect is routing traffic from SSs to open BSs through open RSs. We will show that the problem is computationally hard from both of these aspects.

A graph denoted by $G = (V, E)$ [27] is a collection V of vertices and a set E of edges. An edge is a pair of vertices. A graph is tripartite [3] if and only if the set of vertices of the graph can be partitioned in 3 independent sets. An independent set in a graph is a set of vertices, where now two of these vertices are adjacent. In graph theoretic terminology the network topology for the mentioned problem can be modeled as a tripartite graph. A SS can be adjacent to BSs or RSs. A RS can be adjacent to SSs and BSs. The graph is tripartite, because it is formed by three disjoint sets – set of BS, set of RS and set of SS, where there is no internal connectivity between the elements of any of these sets.

Figure 2.1 presents an example of the WiMAX planning problem with 3 SSs, 2 RSs

Table 2.1: Bandwidth demand, traffic capacity and installation cost in the example problem instance

	Bandwidth Demand	Bandwidth Capacity	Installation Cost
BS-1		50	15
BS-2		50	30
RS-1		40	20
RS-2		35	10
SS-1	10		
SS-2	15		
SS-3	10		

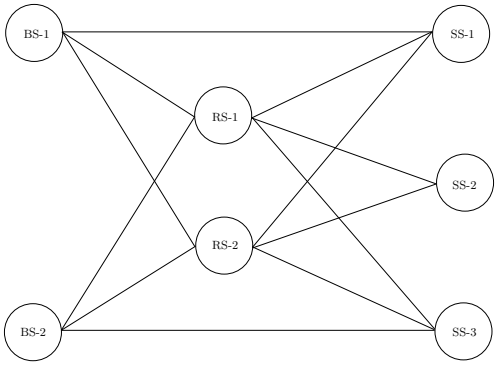


Figure 2.2: Tripartite graph representation of example problem instance

and 2 BSs. The dashed circles represent the coverage areas of the BS/RS in the center. The bandwidth demand for the SSs, and the traffic capacity and installation cost for the RSs and BSs are given in Table 2.1. Figure 2.2 gives the tripartite graph representation of the example problem instance in Figure 2.1.

This problem can be modeled by a minimization problem in a weighted graph $G = (V, E)$, where V represents the set of candidate sites and SSs, E represents the connectivities between the vertices. Let S, R, B be the set of SSs, RSs and BSs, and $V = S \cup R \cup B$. There is an edge between each RS $r \in R$ and each SS $s \in S$, if the channel gain between them is greater than a given threshold. Similarly edges exist between elements of B and S, and elements of R and B depending on the channel gain. Graph G is a tripartite graph because there are no edges between elements of S to elements of the same set, and the same applies for B and R. Each $s \in S$ has a bandwidth demand of d_s. Each of the RSs have a bandwidth capacity limit c_r, it caps the total amount of bandwidth for the SSs connected to it. Similarly, each BS has bandwidth capacity limit c_b that limits the amount of data that can arrive at the node. Each of $r \in R$ and $b \in B$ has establishment cost of f_r and f_b, respectively.

We define the incoming flow at RS $r \in R'$ as $l_i(r)$, and the outgoing flow as $l_o(r)$. Flow $l_i(r)$ is defined as: $l_i(r) = \sum_{s \in S: m_s(s) = r} d_s, \forall r \in R'$. By flow conservation at RS, $l_o(r) = l_i(r)$. By capacity constraint on RS $l_o(r) \leq c_r$. Similarly, the flow at BS $b \in B'$ $l_b = \sum_{m_s(s)=b, s \in S} d_s + \sum_{m_r(r)=b, r \in R'} l_o(r), \forall b \in B'$ by capacity constraint is limited by c_b, i.e. $l_b \leq c_b$. The total installation cost of the network is $F = \sum_{r \in R'} f_r + \sum_{b \in B'} f_b$. A feasible solution to this problem is a subset $B' \subseteq B$ and subset $R' \subseteq R$, and mappings $m_s : S \to B' \cup R'$ and $m_r : R' \to B'$. This mapping has to satisfy constraints involving flow conservation at RSs, capacity constraints at BSs and RSs. Since m_s and m_r are functions, they naturally satisfy the unsplittable flow property. The optimal solution is the minimum F over all feasible solutions.

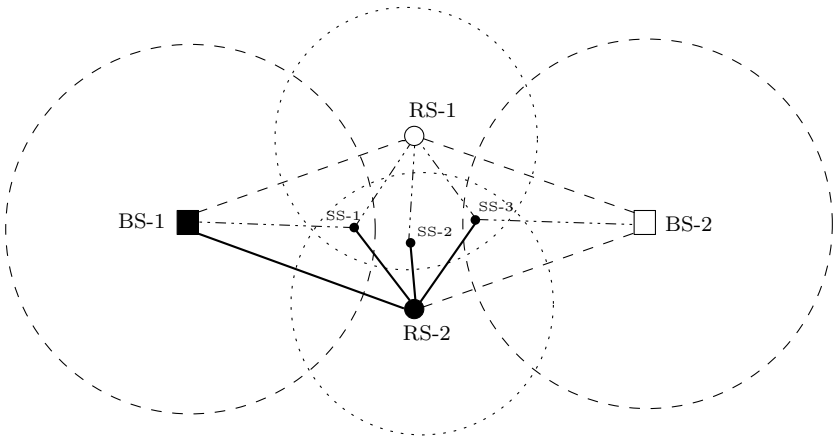


Figure 2.3: Optimal solution for the example problem instance

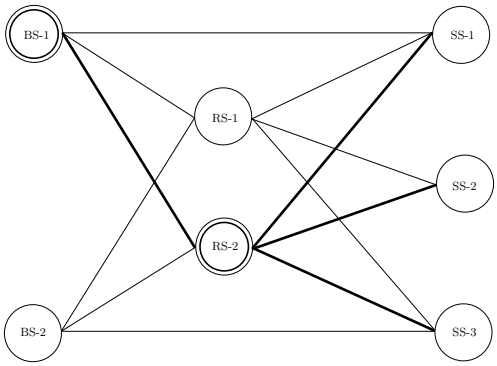


Figure 2.4: Optimal solution in the tripartite graph representation of the the example problem instance

Figure 2.3 provides the optimal solution for the WiMAX planning problem with unsplittable flow in Figure 2.1. The BSs and RSs with filled marks constitute the optimal solution with establishment cost $F = 10 + 15 = 25$.

Figure 2.4 presents the optimal solution in the tripartite graph. The bolded edges represent the used links and the BSs and RSs in double circle represent the selected nodes for the optimal solution. Table 2.2 presents the bandwidth utilization in different BSs and RSs in the optimal solution.

In the following four sections we cover four related optimization problems, which are special cases of the of the WiMAX planning problem. We describe the nature of these

Table 2.2: Bandwidth utilization in the optimal result of the example problem instance

	Traffic Capacity	Bandwidth Utilization
BS-1	50	35
BS-2	50	0
RS-1	40	0
RS-2	35	35

problems, and try to take a look at the solution approaches proposed by different researchers and the quality of the results obtained.

2.2 Bin Packing Problem

The bin packing problem (BP) is an NP-complete combinatorial optimization problem where the primary aim is to pack a finite number of items using the minimum number of bins [10]. Sometimes BP involves other constraints such as capacity, weight, cost, and priority of items, etc. The BP problem has been widely studied due to its various applications in job scheduling [154], industrial balancing [29], computer network design [37], etc. In this section we are looking into the nature of BP problem, its relationship with the WiMAX planning problem, and the research literature available on it.

Given a set $N = \{1,...,n\}$ of items with weights $w_i, i = 1,2,3,...,n$, the BP problem consists of finding the minimum number of k bins of capacity C necessary to pack the items without violating the capacity constraints [10]. In other words, the problem is to partition a set of items into a minimum number of subsets, such that the sum of the weights of the items in each subset is less than or equal to C.

To establish the relationship between the WiMAX planning problem with BP, we simplify the WiMAX planning problem relaxing constraints and making assumptions. We consider the special case where no RS is there in the network, and all the SSs have all the BSs in their transmission range. Here we give a polynomial time reduction of BP problem

to the WiMAX planning problem. Since BP problem is known to be NP-complete, the polynomial time reduction of BP to WiMAX planning problem proves the NP-hardness of the latter problem.

Lemma 1 *The WiMAX planning problem is NP-hard.*

Proof: We use the notation introduced at the beginning of this section and the notation for the WiMAX problem in section 2.1. We construct an instance of the WiMAX planning problem as follows:

1. Let S, be the set of SSs which contain n elements, i.e. $S = \{s_1, s_2, ..., s_n\}$. Let the bandwidth requirement of s_i is w_i, for $i = 1, 2, 3, ..., n$. Let R be empty set. Let B consist of n BSs.

2. Let C be the capacity of all BS.

3. Let all the BSs have installation cost of 1.

4. Let G be a complete[2] bipartite graph.

This reduction can be done in polynomial time. We claim that the BP has a solution in k bins, if and only if the corresponding WiMAX planning problem has a feasible solution with cost at least k. This is because, the k bin solution of the BP corresponds to the subset of BSs with size of the subset as k. Since BP has a polynomial time reduction to a restricted version of WiMAX planning problem, WiMAX planning problem must be NP-hard. This completes the proof of NP-hardness of the WiMAX planning problem. □

Many computer-assisted approaches have been developed to solve BP problems. In 1990 Martello and Toth described a branch and bound procedure widely known as MTP

[2]A complete graph is a simple undirected graph in which every pair of distinct vertices is connected by a unique edge.

[124] which is considered as the basic reference in different research and comparative studies. Another exact method using several bounds, reduction procedures, heuristics, and a branch and bound procedure using a new branching scheme was proposed in 1997 by Scholl et. al. in [140]. Schwerin et. al. [143] showed that MTP provides significantly better results using a bound derived from the cutting stock problem. Valério de Carvalho proposed a column generation and branch and bound based exact algorithm [152]. Valério de Carvalho in 2002 published another paper [153] where he reviewed several linear programming (LP) formulations for BP problems. He analyzed the relations between the corresponding LP relaxations, and their relative strengths, and referred to branching schemes which can be used. The study found Gilmore–Gomory model [76, 77] a better suited LP model to solve BP. Crainic et al. further reduced *optimality gap*[3] on [124] and developed faster methods to obtain more accurate lower bounds [46, 47].

Two of the fastest heuristics for the approximate solution of BP are the well-known First-Fit Decreasing and Best-Fit Decreasing greedy algorithms [124]. In First-Fit Decreasing, the items are first placed in order of non-increasing weight. Then they are picked up one by one and placed into the first bin that is still empty enough to hold them. When no bin is left in which the item can fit in, a new bin is started. In case of Best-Fit Decreasing the best-filled bin[4] that can hold it can be used to fill an item. This makes the algorithm slightly more complicated, but no better. Both heuristics have a guaranteed worst-case performance of $\frac{11}{9}Opt + 4$, where Opt is the number of bins in the optimal solution [42]. Friesen and Langston [68] presented constant factor approximation algorithms for a more general version of BP in which a fixed collection of bin sizes is allowed, and the cost of a solution is the sum of sizes of used bins. Correa and Epstein [45] considered a special version of BP with controllable item sizes, where each item has a list of pairs consisting of

[3] the difference between a best known solution and a value that bounds the best possible solution.

[4] the bin that will have least amount of space left after containing the item.

an allowed size for the item, and a negative penalty associated with it. The goal is to select a pair for each item so that the number of bins needed to pack the sizes plus the sum of penalties is minimized.

Heuristic approaches are very useful for problems, for which deterministic methods are often unable to find the solution within a reasonable time. Kao et. al. described a stochastic approach to solve BP in [97]. In 1994 Hubscher et. al. [89] proposed a tabu search with influential diversification algorithm.

The genetic algorithm community has also been taking interest into BP problem. E. Hooper et. al. [87] reviewed the application of genetic algorithm to the packing problems. It reviewed 2-dimensional packing problems and their solution using genetic algorithm focusing on problem classifications, genetic operators, and other strategies. Earlier Falkenauer [58] described a hybrid grouping genetic algorithm for BP in 1996. It used a grouping approach, the genetic algorithm works with whole bins rather than with individual items. Crossover consists of choosing a selection of bins from the first parent, and forces the second parent to adopt these bins. Any bins in the second parent which conflict with the adopted bins have their items displaced. Local search is used to replace them into the solution: free items are swapped with non-free items to make fuller bins which contain few large items rather than many small items. Any remaining free items are re-inserted into new bins using the First-Fit Decreasing method. Mutation enforces a few random bins to have their items displaced and then re-inserted via the local search procedure. Other genetic algorithms are presented in [53, 59, 91, 137]. Later in 2002 Fleszar et. al. [65] proposed a few new heuristics to BP, the most effective of them being based on the variable neighborhood search (VNS) metaheuristic[5] [82] and using new lower bounds proposed by [61].

[5]Metaheuristics are computational frameworks that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. These algorithms, do not guarantee an optimality of solution.

In 2004 Levine et. al. [112] proposed an ant colony optimization (ACO) [52] approach to solve BP. The paper presents a pure ACO approach, as well as an ACO approach augmented with local search algorithm. Test results showed that the hybrid evolutionary approach could outperform the pure ACO approach for certain problem classes. In the same year Alvim et. al. [7] proposed a hybrid improvement procedure with several features: the use of lower bounding strategies; the generation of initial solutions by reference to the dual min-max problem; the use of load redistribution based on dominance, differencing, and unbalancing, and an improvement process utilizing tabu search. This approach outperformed many of the previous benchmarks, and proved its robustness to perform competitive on all 11 classes of test problems, whereas most approaches reported in the literature could perform well on a subset of the 11.

Meta-heuristics such as simulated annealing [31] and tabu search [122, 145] have been applied to solve higher dimensional BP problems. Evolutionary algorithms [135] including genetic algorithms [93, 120, 161] have also been applied. Most of the evolutionary and genetic algorithms are hybridized with heuristic placement routines. These algorithms search for optimal input sequence of items, and the placement routine determines the packing of inputs in sequence into bins [121]. In 2006 Bansal et. al. [16] proposed an approximation algorithm based on combination of randomized rounding of the optimal solution of LP relaxation to get to a partial integer solution. A more recent study of 2007 Liu et. al. [121] formulated a multi-objective two-dimensional mathematical model with multiple constraints for BP problems. To solve it a multi-objective evolutionary particle swarm optimization [100] algorithm has been proposed.

Another recent work by Baldi et. al. [127] introduced the generalized BP problem, where given a set of items characterized by volume and profit and a set of bins with given volume and cost, aim is to select the subsets of profitable items and appropriate bins to optimize the cost of the used bins and the profit derived by loading the selected items.

The generalized BP problem thus generalizes many other packing problems, including the BP problem, the variable cost and size BP problem [71], Knapsack problem, etc. The paper also presents two mixed integer programming formulations of the problem. One of these yields an efficient column generation based lower bound method, as well as first fit, best fit, and column generation-based upper bound procedures. New instance sets are also introduced and analyzed here, with the results showing that the proposed procedures are efficient and the bounds are tight.

2.3 Set Covering Problem

Set covering (SC) problem [70] is one of the oldest and most widely studied problems in combinatorial optimization. SC problems are computationally difficult optimization problems that have great theoretical significance as well as practical use in areas such as scheduling [14, 11], resource allocation [139], VLSI [6], etc. SC problems arise in different practical situations of operations research, machine learning, planning, data quality and data mining, etc. The SC problem was proved NP-hard [96].

Let U be a set of m elements, and $P = \{p_1, p_2, p_3, ..., p_n\}$ be a collection of subsets of U. Set covering problem is selecting fewest possible subsets from P that include every element in U. U is called the ground set or the set universe. A set cover of U is a subset P' of P satisfying $\cup_{p \in P'} p = U$. In SC problem we intend to find the set cover P' with minimum cardinality.

To establish the relationship between the WiMAX planning problem with SC, we simplify the WiMAX planning problem by relaxing constraints and making assumptions. We restrict to the special case where no RS is there in the network, and all the SSs have a demand of 1 (unit demand). We also assume that all the BSs can serve the demands of all the SSs, which are in their transmission range.

Let U be the set of m SSs and P be the set of n BSs, i.e. $P = \{p_1, p_2, p_3, ..., p_n\}$, where BS p_i has within its communication range a subset of SSs. Let the establishment cost of BS p_i be c_i, for $i = 1, 2, 3, ..., n$. The construction of this special case of the WiMAX planning problem is a reduction to min-cost SC problem. Considering all the BSs have installation cost $c_i = 1$ for i = 1,2,3,...,n, WiMAX planning problem can be further reduced to the SC problem.

The issue of exact resolution of NP-hard problems like SC by algorithms has been actively studied over the last two decades, and a number of optimal algorithms have been experimented. Most of these algorithms are based on tree-search procedures such as branch and bound. Two classical exact solution approaches have been discussed in [15, 20], which can solve small size problems at considerable computational cost. Balas et. al. in [15] report on the implementation and computational testing of an algorithm, based on the cutting planes from conditional bounds, feasible primal and dual solutions, subgradient optimization of a Lagrangian function, and implicit enumeration with branching rules. Beasley et. al. [20] present a 3 step algorithm for the SC problem that combines a dual ascent procedure, a subgradient optimization procedure starting from an initial set of Lagrange multipliers equal to the dual variables from previous step, and solving the dual of the linear programming relaxation of the SC problem.

In 1990 Fisher et. al. [64] presented an optimal solution algorithm based on a dual heuristic. The algorithm could handle SC problems up to an instance where the size of the ground set is 200 and there are 2000 subsets. In the same year Karmarkar [98] proposed an interior-point approach to solve 0-1 integer programming problems. He converted the SC problem to non-convex quadratic programs over polytopes. Experiments using his algorithms with the Steiner triple systems [70] was published in the following year 1991 by Karmakar et. al. [99]. The algorithm produced the best known covers for all test instances, however, they observed that a local optima was encountered during solution and the effect

caused by the local optima was very large running time for large instances of problems. Beasley et. al. [23] combined a Lagrangian heuristic, feasible solution exclusion constraints, and an improved branching strategy to enhance Beasley's previous algorithm [20] and solved problem instances with size of the ground set up to 400 and there were up to 4000 subsets. Harche et. al. [83] developed an exact algorithm, which was capable of solving large sparse instances of SC problems. These optimal solution algorithms are based on tree-search procedures. Harche et. al. named it the Column Subtraction Algorithm. More recent results using exact resolution techniques are presented in [66, 80, 155].

The pioneering research proposing approximation algorithms for SC problem with a worst-case analysis are presented in [95, 123, 41]. These approaches use natural greedy algorithm. This algorithm chooses to include one of the sets of maximum residual cardinality into the solution in every step of its greedy choice. In 1974 Johnson [95] showed that the greedy algorithm gives an approximation ratio of $\ln m$. In 1996 Slavík [147] presented analysis on lower order terms of approximation ratio for the greedy algorithm and showed that the greedy minimum SC algorithm achieved a tighter ratio of $O(\log m)$. Lovász [123] used a linear programming relaxation that approximates SC within a ratio of $\ln m$. In [54], using semi-local optimization techniques, a $\frac{1}{2} + ln|P^*|$-approximation algorithm was given by Duh et. al., here P^* is the maximum cardinality set in P. Other researches around approximation of minimum SC by polynomial algorithms are presented in [86, 79, 132]. Chvatal [41] extended the work of Johnson to the weighted version of SC. In this case the greedy algorithm chooses to include one of the sets, maximizing the ratio between residual cardinality and weight, into the solution in each step of its greedy choice. Analysis of the low-order terms of the approximation ratio for the linear programming approach was provided by Srinivasan [148].

Researchers also used Probabilistically Checkable Proof (PCP) theorem [12] to deal with the quality of possible approximation of SC problem. Feige [60] showed that mini-

mum SC cannot be approximated with an approximation ratio better than $(1-\varepsilon)\ln m$, for every $\varepsilon > 0$. This result exhibits the large gap between quality of approximation achievable in polynomial time and results that can be obtained in exponential time. Bourgeois et. al. [28] presented a review of recent works on approximating minimum cost SC problem. Recently Åstrand et. al. [13] in 2010 presented a distributed algorithm that finds minimum SC. They worked with minimum weight SC problem and proposed an f-approximation algorithm (here f is the maximum frequency of an element). A computational comparative study of different approximation algorithms for SC problem can be found in [81].

In the year 1980 Vasco et. al. [15] presented implementation and computational testing of several versions of a SC algorithm. They used different cutting planes obtained from conditional bounds. The algorithm used a set of heuristics to find prime covers, another set of heuristics to find feasible solutions to the dual linear program which are needed to generate cuts, and sub-gradient optimization to find lower bounds. It also uses implicit enumeration with branching rules. Another heuristic solution procedure for SC is presented in [156]. In 1990 Beasley [21] presented a Lagrangian heuristic algorithm and reported that his heuristic gave better quality results than algorithms in [15, 156], for problems involving up to ground set of size 500 and 5000 subsets.

Later in 1993 Jacobs and Brusco [94] developed a heuristic based on simulated annealing and reported on SC problems with up to ground set size of 1000 and 10000 subsets. In the same year Sen [144] presented results from experiments with a simulated annealing optimization technique for solving the minimum cost SC problem. Experimental results indicated that the algorithms were good to find optimal result for small size problems, and also could find acceptable solutions for mid size instances. The increase in running time of the algorithm with increase in problem size was polynomial.

Some of the research on SC problem used ACO approach. Lessing et. al. [111] in 2004 studied the behavior of different ACO variations to solve SC problem. Ren et. al.

[138] presented an ACO approach to SC problem. They proposed a constraint oriented solution construction method. The construction works as follows: while adding a subset to the partial solution, the algorithm randomly selects an uncovered element from the base set and only considers the subsets covering the element. This process decreases the candidate solution components and accelerates the speed of the algorithm. A local search procedure is incorporated with this. The local search aims at eliminating redundant subsets from the partial solution and replaces subsets with more effective ones while maintaining the feasibility of the solutions. Crawford et. al. in 2009 added a post preprocessing stage with ACO to get better results on SCP problem [48]. Although this improved the solution, it took relatively longer computational time.

Earlier in 2006 Yagiura et. al. presented a 3-flip neighborhood local search method in [160]. The main feature of this local search was the 3-flip neighborhood which is the set of solutions obtainable from the current solution by exchanging at most three subsets. The search was allowed to visit the infeasible regions, and incorporated a strategic oscillation technique realized by adaptive control of penalty weights. The problem size was reduced by using information from the Lagrangian relaxation, this was very important to solve very large instances.

Liepins et al. [114, 115] investigated genetic algorithms for SC problems with two types of crossover operators in conjunction with three penalty functions and two multi-objective formulations. Their results are encouraging and point to the greedy crossover, and tight upper bounds for cost of completion of covers (as a penalty function). Feo et. al. [62] pursued a non-deterministic method for solving SC problem. The procedure is based on Chvatal's greedy approach [41]. In order to improve upon Chvtal's approach they introduced randomization. Huang et. al. [88] introduced a genetic algorithm based approach for SC problems. The paper described the procedure for generating a new penalty function to produce better results. In addition it demonstrated that uniform crossover is more efficient

than the greedy crossover. The paper also proposed a mutation operator that can approach the optima from both sides of feasible/infeasible borders, and thereby accelerates the convergence to the optimal solution. Huang's approach performed better than the results of Feo. et. al. [62]. In 1996 Basely et. al. [22] proposed a genetic algorithm-based heuristic for non-unicost SC problems. The proposal had several modifications to the basic genetic procedures including a new fitness-based crossover operator (fusion), a variable mutation rate and a heuristic feasibility operator tailored specifically for the SC problem. The performance of the algorithm was evaluated on a large set of randomly generated problems. Computational results showed that the the heuristic is capable of producing high-quality solutions.

Some of the researchers considered SC problem for very large data sets, especially when the data is disk resident. Berger et al. [24] was concerned with parallelizing the heuristic for SC problem. They allowed multiple computing entities with shared memory to choose sets at the same time. This work also used randomization to ensure that multiple processors do not pick sets which cover the same elements redundantly. This algorithm assumed full random access to the memory for all the processors. A very recent study in 2010 by Cormode et. al. [44] observed that the natural greedy algorithm for SC problem, that picks up the set with largest number of uncovered items at every step, does not work for very large data sets. This is because the natural greedy algorithm makes random access to the disk, which is costly in comparison to linear scans. This paper proposes an algorithm which finds a solution close to that of greedy, and is more efficient to implement using modern disk technology.

2.4 Capacitated Set Covering Problem

In this section we consider the set covering problem with capacity constraints, the Capacitated Set Covering (CSC) problem [34]. CSC problem can be seen as a generalization of the bin packing problem that include several types of bins. Given a set of elements U, each element with its demand, a set P of subsets of U (types of bins), and an upper bound on subset capacity, the objective is to partition the elements into a minimum number of copies of the subsets (bins) so the total demand of elements assigned to each set copy does not exceed the upper bound on set capacity. The CSC problem is an NP-hard problem [34], and is a natural generalization of a real world problems that capture practical scenarios where resource limitations are present. As an example, the goal in the minimum Steiner tree problem [90] is to find the minimum cost subgraph connecting terminals to a root. We can consider this as a set cover problem. To show that we consider each graph cut separating a subset of terminals from the root as an element, and each edge in the graph is considered a set that covers every cut that the edge crosses. With the terminals having bandwidth requirements, and the edges having capacities, this corresponds to a CSC problem.

Let $U = \{1,2,3,...,n\}$ be a ground set of elements, and let P be a collection of sets defined over U (i.e. $P \subseteq 2^U$, with 2^U the set of all subsets of U). Each $P' \in P$ has a non-negative cost $w(P')$ associated with it. A cover C is a collection of sets such that their union is U.

To define CSC, Carr et. al. [34] assumed that each set $P' \in P$ has a capacity $k(P')$ associated with it, meaning that it can cover at most $k(P')$ elements. The goal was to find a cover of minimum cost that maintains the capacity constraints.

CSC problems are basically two kinds: *soft capacities problem* and *hard capacities problem*. In case of soft capacities, an unbounded number of copies of each covering object is available. For hard capacities each covering object ($P' \in P$) has a bound $m(P')$ on

the number of available copies. Thus, a cover C is a multiset of input sets that can cover all the elements, while C contains at most $m(P')$ copies of each $P' \in P$, and each copy covers at most $k(P')$ elements. [40]

Bansal et. al. [18] perceived CSC problem in a slightly different way. Here the elements of U have demands $d : U \to \mathbb{R}^+$, each element in P have supplies $s : P \to \mathbb{R}^+$. The goal here is to find a minimum cost subset $P' \subset P$ such that for each element e, the total supply of sets in P' that cover e is at least $d(e)$.

To establish the relation between WiMAX planning problem with CSC (as defined by Bansal et. al. [18]), we simplify the WiMAX planning problem to a special case where no RSs are there in the network. We construct an instance of the WiMAX planning problem as follows:

1. Let U be the set of SSs with demand $d(e)$ for all $e \in U$.

2. Let R be empty set.

3. Let P be the set of BSs with non-negative installation cost $w(P')$ and capacity $k(P')$ for all $P' \in P$. BS $P' \in P$ can serve a subset of SSs $U_{P'} \subseteq U$.

As per definition of CSC, the above construction on an instance of WiMAX planning problem produces an instance of CSC with bound on multiplicity $m(P') = 1$ on the number of available copies of $P' \in P$.

In 1981 L. Wolsey published a research [159] on submodular[6] set covering problems and proposed an approximation algorithm. This algorithm used the classical greedy algorithm in [41] for set covering problem. It selects a set whose addition to the cover maximizes the ratio of the increment of processed demand to the set cost. Thus, the approximation factor relies on k, as the maximum possible increment of processed demand

[6]Let f be an integer valued function defined over all subsets of a finite set of elements E. Function f is *submodular* if $f(S) + f(T) \geq f(S \bigcap T) + f(S \bigcup T)$ for all $S, T \subseteq E$.

due to the addition of a single set to the cover. Thus Wolsey's proposed algorithm is an H_k approximation algorithm for CSC problems, where $H_k = 1 + \frac{1}{2} + ... + \frac{1}{k}$, k is the maximum capacity of a set $P' \in P$. This algorithm is applicable to both the hard and soft capacity versions of CSC. Later in 2007 Alfandari [5] was studying capacitated facility location problem with soft capacities and proposed an $(1+\varepsilon)H_n$-approximation algorithm for any $\varepsilon > 0$, n is the number of elements in the universal set U. Since this algorithm does not consider the edge distances, it can be used for CSC problem with soft constraints.

Earlier in 2000 Carr et. al. [34] investigated the usefulness of a standard integer programming (IP) formulation and its linear programming relaxation (LP) for finding good approximations of capacitated covering problems. They observed that the ratio between integer optimal solution and optimal solution to the linear programming relaxation can be arbitrarily large for capacitated covering problems. It is because of this fact that a good approximation guarantee cannot be obtained from the linear programming relaxations of capacitated cover problems. Carr et. al. proposed introduction of a new class of simple inequalities to the linear program, called the *flow cover inequalities*, to strengthen the linear programming relaxation. They showed that for particular instances of capacitated covering like minimum knapsack and capacitated network design, that these inequalities could yield better IP/LP ratios. Carnes et. al. [33] used flow cover inequalities proposed by Carr et. al. to develop a primal-dual schema for capacitated covering problems. The primal-dual algorithms by Carnes et. al. was tested on three special cases of CSC – *Minimum Knapsack Problem*, *Single-Demand Facility Location Problem* and *Single-Item Lot-Sizing Problem*. These algorithms could achieve a worst case performance guarantee of 2. More recent works on capacitated covering problem can be found in [40, 57, 72].

In 2010 Berman et. al. [25] proved that CSC problems can be approximated by a polynomial time algorithm with approximation ratio $r + 1.357$ if the underlying uncapacitated version can be approximated with ratio r by the same algorithm. In the same year

Chakrabarty et. al. [35] considered the correlation between CSC, and the underlying standard set cover problem. They tried to exploit information from the matrix A to make a good approximation to the CSC problem. Chakrabarty et. al. proposed to reduce the approximability of CSC to the problem of bounding the integrality gaps of the standard LP relaxations of 0,1-covering problems. They considered two classes of such problems. The first is the *Underlying 0,1-Multicover Problem*, which is the family of weighted set multicover[7] instances defined by matrix A. The second is the *Priority Covering Problem* that is obtained from the CSC instance, and priorities $\pi_{P'}$ and π_e for all sets $P' \in P$, elements $e \in U$, and $\pi : U \cup P \Rightarrow Z^+$. In this case a set P' covers element e, if $e \in P'$, and if $\pi_{P'} \geq \pi_e$. Chakrabarty et al. showed that if α and β are upper bounds on the integrality gaps in the *Underlying 0,1-Multicover Problem* and *Priority Covering Problem* respectively, the CSC has a $O(\alpha+\beta)$ approximation for the given instance.

A question left open in [35] was the significance of matrix A being a network matrix[8]. In 2011 Bansal et al. [17] addressed this specific case and proposed an $O((log\ log\ s_{max})^2)$ approximation to the CSC problem, where s_{max} is the largest supply. Another very recent work by Chan et. al. [36] improved over the work by Bansal et. al. and proposed an $O(log\ log\ N)$ approximation for CSC, where N is the number of elements in the universal set.

In another study Chang et. al. in 2011 [38] considered a special case of the CSC problem with maximum cardinality of a set fixed to 2. Interestingly for this problem the corresponding uncapacitated set covering problem is the maximum matching problem, which is polynomially solvable. Chang et. al. proposed a 1.5 approximation algorithm for this

[7] the usual generalization of standard set cover where an element may wish to be covered by several distinct sets, instead of just one.

[8] Let $G = (V,E)$ is a directed graph[9] and let $T = (V,E')$ be a directed tree. Let C be the $E' \times E$ matrix defined as follows. Take $a' \in A'$ and $a = (u,v) \in A$ and let P be the undirected $u-v$ path in T. Define $C_{a',a} :=\{$ +1 if a' occurs in forward direction in P, -1 if a' occurs in backward direction in P, 0 if a' does not occur in P.$\}$ Matrix C is called *network matrix* generated by $T = (V,E')$ and $G = (V,E)$ [142].

special case of CSC.

2.5 Unsplittable Flow Problem

In the unsplittable flow problem (UFP) [103], given a graph with edge capacities and a set of source nodes and destination nodes, and bandwidth requirements associated with each of the destination nodes, we are supposed to find a set of source-destination paths to satisfy the demands in the destinations such that there is exactly one path between a source and a destination. The demand of each destination must be routed so that the total flow through any edge is at most its capacity. The single source UFP was introduced by Kleinberg [103] in 1996. In this problem, data must be routed simultaneously from the one source node to the destinations in the given graph.

UFP is a generalization of edge-disjoint path problem [103], where every edge has a positive capacity, and every commodity has a demand which should be routed in an unsplittable manner. It is an NP-hard variant of maximum flow problem [103]. If we relax the constraints that each source-sink flow should use exactly one path, and if all sources coincide at a vertex and all the sinks at a vertex, it reduces to the maximum flow problem which is solvable in polynomial time. The problem can also get reduced to other polynomial time solvable problems like minimum cost flow, if there is cost associated on the edges, depending on the objective.

The WiMAX planning problem prohibits the splitting of flow between SSs and BSs, i.e. data in this network flows through no more than one path between a SS and the BSs. This nature of traffic relates the problem in hand with the UFP. In this section we study single-source case of UFP, the different optimization variants of it, and try to see what approximation and heuristic approaches have been applied on the problem.

Let $G = (V, E)$ be a directed graph with edge capacities $u : E \rightarrow R^{+}$ and cost function

$c : E \rightarrow R^+$, a designated source vertex $s \in V$ and k commodities each associated with a terminal t_i and a demand $d_i \in R^+$, $1 \leq i \leq k$. Here R^+ is the set of positive real numbers. The single source UFP asks for a feasible unsplittable flow which can route d_i units of commodity i along a single $s - t_i$ path for each i without violating the capacity constraint of any edge $e \in E$. Here we use $T \subseteq V$ to denote the set of k terminals, $|T| = k$, $|V| = n$, $|E| = m$ [102].

An unsplittable flow f contains a set of paths $\{P_1, P_2, ..., P_k\}$, where P_i starts at the source s and ends at t_i. The cost of a path P_i is defined as $C(P_i) = \sum_{e \in P_i} c_e$ and the cost $c(f)$ of a flow f is given by $c(f) = \sum_{e \in E} c_e f_e$, (with f_e as the total flow on edge e, i.e. $\sum_{e \in P_i} d_i$).

From the available literature [103, 51, 105] we can identify four main optimization versions of the problem:

1. *Maximization:* What is the routable subset of terminals of maximum total demand? In other words, we are to find a subset of $T_1 \subseteq T$ which can be routed unsplittably such that $\sum_{i \in T_1} d_i$ is maximized?

2. *Routing in Rounds:* What is the minimum number of rounds we can partition the terminals into, such that terminals in the same round are unsplittably routable?

3. *Congestion:* What is the smallest value by which we can multiply all the capacities to satisfy all the demands?

4. *Minimum Cost:* What is the least flow cost to route all the commodities with unsplittable flow?

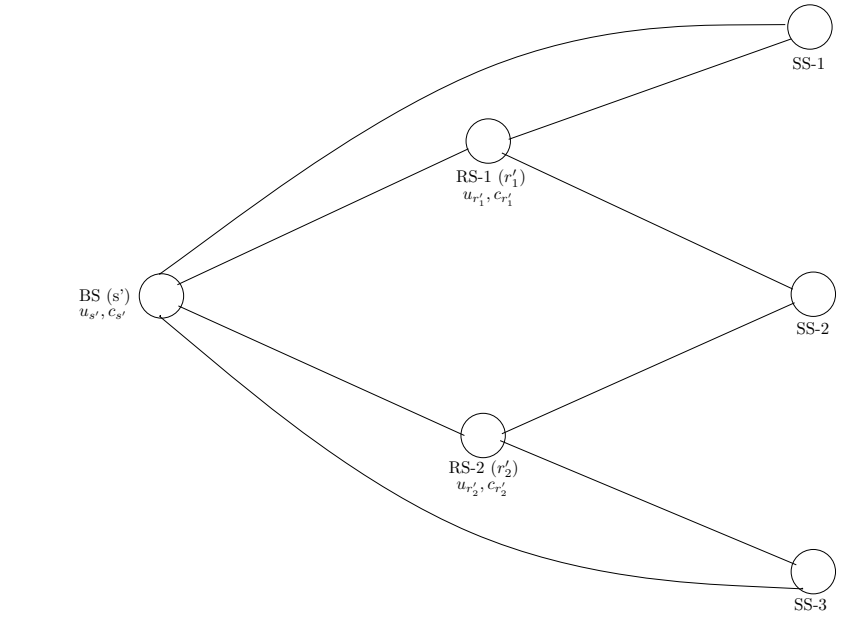
To establish a relation between the WiMAX planning problem with UFP, we consider minimum cost single source UFP first. Given an instance of WiMAX planning problem, we construct an instance of UFP. We simplify the WiMAX planning problem and restrict it to a special case where there is only one BS s' in the network. Let T be the set of k SSs

with demand d_i associated with SS t_i for $i = 1, 2, ..., k$. Since UFP has capacities and costs on the arcs of the network, we will replace the BS and each of the RSs in the network as a couple of vertices connected by a directed edge. We associate the capacity and cost of the BS and RSs on this new edge. Let r' be any RS connected to BS s' and set of SSs T'_r. In the new graph RS r' will be represented as adjacent vertices r and r_{in}, and a directed edge with capacity u_r and cost c_r on the edge from r_{in} to r. The value of u_r and c_r will be the same as capacity and cost on RS r'. Each SS $t \in T_r$ has a directed edge towards r_{in}, the capacity of this edge is set to $+\infty$ and the cost is set to 0. r has a directed edge towards BS s' with capacity and cost on the edge set to $+\infty$ and 0 respectively. The same is done for BS s'. Figure 2.5(a) presents an instance of WiMAX planning problem, and Figure 2.5(b) presents the UMF relaxation Figure 2.5(a).

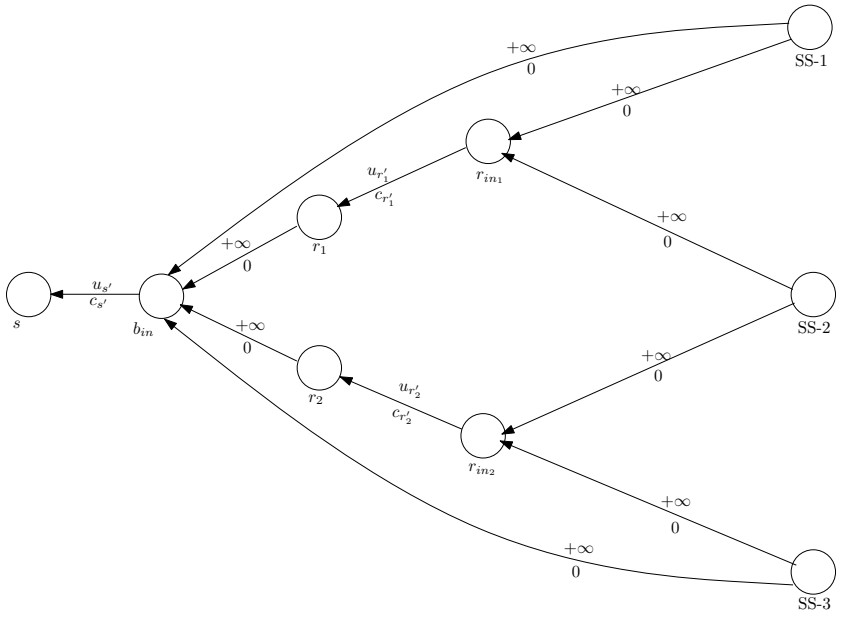
Following the definition of UFP, if there is a unsplittable flow f with cost $c(f)$ satisfying all the demand of the elements of T in the network instance constructed above, the corresponding instance of WiMAX planning problem has a feasible solution with the same cost. The converse is also true.

We can extend the above construction to a case where there is more than one source $S \subset V$ in G, and the unsplittable flow has to ensure a single path between $s_i \in S$ to all $t_i \in T$ to satisfy all d_i. This is a relaxation of WiMAX planning problem where we allow RSs to connect to multiple BSs, however they have to route the demand for a particular SS through exactly 1 BS.

The first constant factor approximations for single source UFP were given by Kleinberg [103, 102]. He assumed that the minimum of edge capacities is greater or equal to the maximum demand (the no bottleneck assumption) and scaled the demands and capacities such that all capacities are at least 1 and all demands are at most 1. The algorithms presented here are based on rounding methods for converting fractional flows into unsplittable flows. The fractional flow for single source UFP is the classical max flow problem [67] or



(a)



(b)

Figure 2.5: Example of transforming WiMAX planning problem to UFP

min cost flow, depending on the objective. Kleinberg's algorithm finds a set of trees in G such that every node of G is in at least one such tree. Such set of trees is called the *tree cover* of G. Every tree in this *tree cover* set contains the amount of demand corresponding to the terminals that are in the tree. These trees are subgraphs of G with terminals as the leaves of the tree. Each of these trees in *tree cover* set also have a small subset of edges in G. Such a *tree cover* can be obtained from G by running a depth first search on a spanning tree of G, while making sure that the total demand in a group is within predefined bounds. The root node from each of these trees is chosen (as leader), and a path is found from this (leader) node to the source. Thus the terminals in each tree are routed to their leader using the edges of the tree and concatenating this path with the path from the leader to the source the unsplittable flow is obtained. The demand maximization algorithm routes $(1-O(\sqrt{d_{max}})OPT_d$, where OPT_d is the optimal value for version concerning with maximizing total demand. Kleinberg's algorithm routes all demands unsplittably with congestion at most $(1+O(\sqrt{d_{max}/OPT_c})OPT_c$. Here OPT_c is the optimal value for the congestion minimization version. If $d_{max} < 7/4 - \sqrt{3}$ and a maximum fractional flow exists, the total demand can be unsplittably routed in 2 (two) rounds. Kleinbuerg gave 16-approximation algorithm for minimum congestion in directed graphs, and 8.25-approximation in the undirected case.

Kolliopoulos et. al. [106, 104, 105] also adapted the no bottleneck assumption of Kleinbuerg. Algorithms presented in [105] first found the max flow. Information from max flow is then used to allocate capacity to different subproblems, where each subproblem contains demands in a fixed range. A separate near-optimal solution to each subproblem is found and the solutions are then combined. This paper presented a $(3.23+o(1))$-approximation for both directed and undirected graphs in the case of minimum congestion. They give 1.68 approximation algorithm for congestion with a simultaneous performance guarantee $3.23+o(1)$ for cost denoted by $(1.68, 3.23+o(1))$ simultaneous approximation on

both directed and undirected graphs. The best known existing result [103] at that time for undirected graphs was simultaneous (7.473,10.473) approximation for cost and congestion. Concerning maximum routable demands, [105] showed that their algorithm could route at least .075 of the optimum. Concerning the number of rounds, [105] showed how to route all the demands in at most 13 times the optimum number of rounds. The work in [106, 104] are developments on the research presented at [105]. Kolliopoulos et. al. in [106] gave a (3, 2)-approximation for congestion and cost.

In 1999 Dinitz et. al. [51] assumed the no bottleneck condition, and another assumption called the *cut condition*. The *cut condition* states that for any set S of vertices of G that does not contain the source s, the total demand of the terminals within S is at most total capacity of the edges entering S. For the congestion minimization problem, they obtain an unsplittable flow where the increase in congestion is at most d_{max}. Dinitz et. al. showed that the congestion on any edge a is at most $f_a + d_{max}$, given fractional flow f satisfying all demands in an instance where the no bottleneck condition holds. The assumption that all capacities are at most 1, gives an unsplittable flow with congestion at most 2. Thus Dinitz et. al. improved the congestion bound to 2. They showed that any splittable flow satisfying all demands can be turned into an unsplittable flow while increasing the total flow through any edge by less than the maximum demand [51]. Their algorithm had a congestion rate of $(3+2\sqrt{2})$, while assuming that there might be bottleneck on some edges. Concerning number of rounds, they showed that all the demands can be routed unsplittebly in 5 rounds. Concerning maximum routable demands, their algorithm can satisfy 22.6% of the total demand by unsplittable routing. Dinitz et. al. extended results to the case when the *cut condition* is not satisfied. For this case they derived a 2-approximation algorithm for congestion, 5-approximation algorithm for number of rounds, and 4.43- approximation for maximum routable demand.

Earlier in 1990, Lenstra et. al. [110] show that the minimum congestion problem cannot

be approximated within less than $3/2$, unless $P = NP$. Another negative result was due to Erlebach et. al. [55], who prove that for arbitrary $\varepsilon > 0$ there is no $(2-\varepsilon, 1)$-approximation algorithm for congestion and cost unless $P = NP$.

In 2002 Skutella [146] improved the $(3,2)$-approximation algorithm for congestion and cost of Kolliopoulos et. al. [106, 104] to a $(3,1)$-approximation algorithm. Skutella also assumed that no edge in his instance graphs could be a bottleneck, i.e. $min\ u_i \geq max\ d_i$. He named the instances that satisfied this assumption as *balanced instances*, and the violating ones as *unbalanced instances*. Instances in which maximum demand is ρ times the minimum capacity for $\rho > 1$, are $\rho - unbalanced$. Skutella showed that, unless $P = NP$, congestion cannot be approximated within less than $(1+\sqrt{5})/2$ for the case of $(1+\sqrt{5})/2$-unbalanced instances. Let $f_i^j(e)$ is the flow on edge e. The algorithm here starts with a fractional flow f_0^1 of minimum cost satisfying all demands. If the algorithm ever finds an edge with 0 flow, it is removed. Otherwise, it considers the demands in non-decreasing order. At each iteration i, the terminal with the next smallest demand d_i is processed. The flow from the previous iteration f_{i-1}^1 is used to set new edge capacities u_e^i. Thus, a d_i-integral feasible flow f_i^0 with cost at most the cost of the previous flow is obtained. This flow is d_i-integral and satisfies all demand, therefore there is at least one path P_i from s to t_i and all edges on this path have at least d_i flow. Hence, the demand of terminal t_i can be routed along P_i and the flow on this path is then decreased by d_i. The flow at the end of this iteration is denoted by f_i^1. The algorithm ends with a set of paths $P_1, P_2, ..., P_k$ along which demand for each terminal can be routed from the source. The flow on any edge e is at most $f_0^1(e) + d_{max}$ and the cost of the flow is less than the cost of f_0^1.

More recently in 2007, Peng et. al. [133] considered the minimum cost version of single source UFP, and proposed $(3+2\sqrt{2}, 1)$-approximation algorithms for minimizing congestion and cost.

Some of the works mentioned in the literature applied branch and bound based exact

methods to solve UFP. Pióro et. al. [136] reported that application of branch and bound based methods on UFP is limited to at most medium sized networks (around 20 nodes). This is why the heuristic methods seems to be promising to get a good solution in reasonable processing time. Greedy algorithm (GA) has been the most intuitive approach to solve the UFP. The natural greedy algorithm [102] for UFP processed all connections in one pass and either allocate the processed request to the shortest path or reject the request if such feasible path does not exist. A modification of greedy called bounded greedy algorithm (BGA)[107, 102] works in the following steps: Let L be a suitable chosen parameter. Reject the request if there is no feasible path of the length at most L hops. Otherwise accept the request. Another version of GA, careful BGA (cBGA)[107], orders the requests according to their demands starting with the heaviest. cBGA accepts a request if there exists a feasible path for the request such that after routing the request the total flow on at most $\sqrt{|E|}$ edges of the path is larger than half of their capacity, where $|E|$ is the total number of edges in the input instance. Walkowiak [158] proposed two new heuristic algorithms to address the unsplittable flow problem: Greedy Algorithm with Preemption (GAP) and Greedy Algorithm with Preemption and Flow Deviation (GAPFD). These algorithms can be applied for flow optimization in networks of various size and topology. The key idea of their algorithms centers around the preemption and re-optimization of already established connections. In this case preemption is done by removal of already established connections from the network to enable establishment of other connections. This paper also presents result from numerical experiments, where performance of existing heuristics [107, 102] have been compared with algorithms proposed. Reported results show that GAPFD out performs other algorithms.

A generalization of UFP is multiple sources unsplittable multicomodity flow problem, where there is a set of sources $S = \{s_i,\ i > 1\}$ instead of a single source s and demands are requested between terminal pairs (s_i, t_i). The unsplittable flow solution in this case

has paths P_i connecting each of the $t_i \in T$ to exactly one of $s_i \in S$. These problems are also found in different practical world scenarios and mentioned in literature as unsplittable multicomodity problem, this class of problems is proven to be NP-hard [39]. Unsplittable multicomodity problem has been investigated by researchers with different objectives such as minimizing cost [113], minimizing congestion [26], etc. In 2000 Barnhart et al. [19] proposed an exact method for this problem, the limitation though is it cannot handle instances beyond small size networks. Approximation algorithms [39, 49] have also been used to solve the problem. In 2009, a meta heuristic approach based on ACO was proposed [26]. Masri et. al. [125] in 2011 studied the single path multi-sources multi commodity communication flow problem (MMCF) in the context of a computer network. Messages are to be routed in a capacitated network including a set of source nodes and terminals. Each edge in the network is characterized by a capacity, a transmission delay and a cost. Masri et. al. proposed a mathematical formulation of the MMCF as a 2-objective optimization problem that minimizes the overall cost and delay. Structural constraints are respected as the capacity on the edges and the single path for routing messages. A solution of the proposed model provides for each request the assigned source node, the transmission path as well as the bandwidth allocated along the path. They proposed an ACO meta heuristic to solve the problem. The algorithm implements an iterative reverse path construction strategy so that the ants start from the terminals and move until reaching the source. At each iteration, a uniobjective nonlinear subproblem is solved to optimize the bandwidth allocation for each generated path, so that the transmission rates of the sources will be set by the routing algorithm [125]. Masri et. al. also proposed a lower bound approximating the total delay.

2.6 Summary

In this chapter we have reviewed the available literature on four classical NP complete problems – Set Cover, Capacitated Set Cover, Bin Packing and Unsplittable Flow Problem. We have established that restricted versions of our research problem is polynomially reducible to these problems, and thereby also NP complete. No polynomial time algorithms are known to solve NP complete problems, the same applies to our research problem. If the problem instances have few BS, RS and SS, we can exhaustively enumerate the possible solutions and pick the best result. But when the input instances have large number of BSs, RSs and SSs, exhaustive search for the best solution would take exponentially large time with respect to the input size.

We have studied how different approximation and heuristic algorithms were applied on Set Cover, Capacitated Set Cover, Bin Packing and Unsplittable Flow Problem in Section 2.2 - Section 2.5 of this chapter. We are convinced on applying heuristics/meta heuristics based solution space search to approximate a near-optimal result in reasonable computational time and resource.

In the light of literature reviewed in the chapter, we have implemented 2 variations of local search and variable neighborhood search based heuristic approaches. They are presented in Chapter 3. We have also formulated the WiMAX planning problem as a set cover instance, and have applied Lagrangian relaxation based heuristic approaches to get lower bounds on the problem, they are presented in Chapter 4.

Chapter 3

A Combinatorial Approach to Solve WiMAX Planning Problem

The first approach that we have taken towards solving the WiMAX planning problem is making mathematical formulations of the problem and using off the shelf mathematical programming solver software to obtain solutions. The intention is to see how good currently available software can solve these problems. In this chapter we present two mathematical formulations of the WiMAX planning problem, and compare them. In chapter 5 we have presented a third formulation based on column generation and used Lagrangian relaxation on the model. We have used mathematical problem solver software ILOG CPLEX 12.1.0 [92] and observed their long running time for different instances. This motivated us to design algorithms that would be able to quickly produce near optimal results. We present few variations of greedy heuristics on the problem. We also present few variations of local search on the problem. We also propose a variable neighborhood search framework that we have implemented and tested. All the heuristic algorithms described in this chapter have graph $G = (V, E)$ as input. All these algorithms intend to produce a solution $A \subseteq (R \cup B)$. The experimental results using these algorithms are presented in Chapter 5.

3.1 Modeling the Problem

In this section we present two integer programming (IP) formulations through which we can express the WiMAX planning problem. Both of these have the same objective function, but they differ in modeling the constraints. The first one has discrete variables on the vertices and edges, and variables representing the flow on the connections between RSs and BSs. This formulation has capacity constraints, flow conservation constraints, and unsplittable

flow constraints. The second formulation has discrete variables on the RSs and BSs, and on all the paths between SSs and BSs. This model has capacity constraints and unsplittable flow constraints. However, it does not have explicit flow conservation constraint.

3.1.1 Model Based on Flow Conservation and Capacity

Let S be the set of SSs, R be the set of RSs and B be the set of BSs that can be used to satisfy demands of elements in S. Let d_s be the demand of SS $s \in S$, f_r and f_b be the installation costs for $r \in R$ and $b \in B$ respectively. Let c_r and c_b be the bandwidth capacity for RS $r \in R$ and BS $b \in B$ respectively. Let S_r be the set of SSs within the communication range of RS r, and R_b be the set of RSs within the communication range of BS b and S_b be the set of SSs within the communication range of BS b. We define variables: y_r, y_b corresponding to each of the RS and BS, x_{rs} corresponding to the SS-RS edges, x_{bs} corresponding to the SS-BS edges, x_{br} corresponding to the edges between BS-RS connections, and z_{br} corresponding to the flow through RS r to BS b. The variables are described as follows:

$$y_r \in \begin{cases} 1 & \text{if } r \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

$$y_b \in \begin{cases} 1 & \text{if } b \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{rs} \in \begin{cases} 1 & \text{if } s \text{ is served by } r \\ 0 & \text{otherwise} \end{cases}$$

$$x_{bs} \in \begin{cases} 1 & \text{if } s \text{ is served by } b \\ 0 & \text{otherwise} \end{cases}$$

$$x_{br} \in \begin{cases} 1 & \text{if } r \text{ is served by } b \\ 0 & \text{otherwise} \end{cases}$$

z_{br}: the total demand of r served by base station b

The problem of satisfying the SSs' demands by minimum cost installation of RS and BS can be formulated as follows:

$$Minimize \sum_{r \in R} y_r f_r + \sum_{b \in B} y_b f_b$$

Subject to:

$$\sum_{r \in R} x_{rs} + \sum_{b \in B} x_{bs} = 1, \forall s \in S \tag{3.1}$$

$$y_r c_r \geq \sum_{s \in S_r} x_{rs} d_s, \forall r \in R \tag{3.2}$$

$$\sum_{b \in B, r \in S_b} z_{br} = \sum_{s \in S_r} x_{rs} d_s, \forall r \in R \tag{3.3}$$

$$\sum_{b \in B, r \in S_b} x_{br} = y_r, \forall r \in R \tag{3.4}$$

$$c_r x_{br} \geq z_{br}, \forall r \in R, b \in B : r \in S_b \tag{3.5}$$

$$y_b c_b \geq \sum_{r \in R_b} z_{br} + \sum_{s \in S_b} x_{bs} d_s, \forall b \in B \tag{3.6}$$

The objective is to minimize the total establishment cost of BSs and RSs. Constraint (3.1) ensures that each of the SSs are served by exactly one RS or BS. Equation (3.2) is the capacity constraint for RSs and ensures that no RS can take more inflow than its capacity. Equations (3.3) and (3.5) makes the flow conservation constraint on RSs, it ensures that the summation of all the outflows from a RS must be always greater than or equal to the inflow of the RS. Constraint (3.4) ensures that each of the open RSs are served by exactly one BS. Equations (3.6) is the capacity constraint on BSs, it makes sure that the total incoming flow to a BS from its adjacent RSs and SSs are less than or equal to its capacity.

3.1.2 Model Based on SS-BS Path and Capacity

We adapt the same notation for coverage area, capacity and installation cost of RS and BS, and demand of SS, as we assumed in the previous model. We define variables: y_r, y_b corresponding to each of the RS and BS, x_{srb} corresponding to the path from SS s to BS

b going through RS r, x_{sb} corresponding to the SS-BS paths that does not go through any relay, x_{rb} corresponding to the paths between RS r and BS b. The variables are described as follows:

$$y_r \in \begin{cases} 1 & \text{if } r \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

$$y_b \in \begin{cases} 1 & \text{if } b \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{srb} \in \begin{cases} 1 & \text{if } s \text{ is served by } b \text{ through } r \\ 0 & \text{otherwise} \end{cases}$$

$$x_{sb} \in \begin{cases} 1 & \text{if } s \text{ is served directly by } b \\ 0 & \text{otherwise} \end{cases}$$

$$x_{rb} \in \begin{cases} 1 & \text{if } r \text{ is served by } b \\ 0 & \text{otherwise} \end{cases}$$

The problem of satisfying the SSs' demands by minimum cost installation of RS and BS can be formulated as follows:

$$Minimize \sum_{r \in R} y_r f_r + \sum_{b \in B} y_b f_b$$

Subject to:

$$\sum_{r \in R_s,\ b \in B_r} x_{srb} + \sum_{b \in B_s} x_{sb} = 1, \forall s \in S \tag{3.7}$$

$$y_r c_r \geq \sum_{s \in S_r,\ b \in B_r} x_{srb} d_s, \forall r \in R \tag{3.8}$$

$$y_b c_b \geq \sum_{r \in R_b,\ s \in S_r} x_{srb} d_s + \sum_{s \in S_b} x_{sb} d_s, \forall b \in B \tag{3.9}$$

$$x_{rb} \geq x_{srb}, \forall r \in R,\ s \in S_r,\ b \in B_r \tag{3.10}$$

$$\sum_{b \in B_r} x_{rb} = y_r, \forall r \in R \tag{3.11}$$

$$y_b \geq x_{rb}, \forall b \in B,\ r \in R_b \tag{3.12}$$

As it was in the previous model, objective function for this model is to minimize the total establishment cost of BSs and RSs. Constraint (3.7) ensures that each of the SSs are served by exactly one RS or BS. Equations (3.8) and (3.9) are the capacity constraint for RSs and BSs, they ensure that no RS or BS can take more inflow than their capacity. Equation (3.11) is the unsplittable flow condition on relays making sure that a RS can be served by exactly one BS. Constraint (3.10) enforces that if a subscriber is getting served by BS b through RS r, than RS r is also getting served by BS b, however RS r is getting served by BS b does not necessarily mean all SS $s \in S_r$ are getting served through this path. Equation (3.12) states that a RS r can be served by BS b only if the BS is open, but not the other way around, meaning that an open BS b is not necessarily serving all its adjacent RS $r \in R_b$.

3.1.3 Comparison Between the two Models

Both these two formulations have been tested on small size instances of the WiMAX planning problem, and produced the same optimal solutions. The second model has constraint (3.10) and (3.12), which generates one constraint for each of the paths between SSs and BSs going through RSs, and RSs and BSs. When the WiMAX planning problem has a lot of RSs and BSs in a small area (meaning a dense graph), the total number of such paths are large. We figured while trying to run large instances of the problem the second instance exhausted the memory of the computer we have been running the experiments on. To be more specific, any instances larger than 2000 nodes (1200 SSs, 500 RSs, 300 BSs) crashed in the computer we were running experiments on. The first model, on the other hand, could run with larger instances. Because of this fact, we used the first model presented in Section 3.1.1 for our experiments. In the rest part of the chapter we will be referring to the formulation in Section 3.1.1 as the IP model for WiMAX planning problem.

We have used a numerical optimization problem solver software ILOG CPLEX 12.1.0 [92] to run the IP models for different instances of the WiMAX planning problem. We observed that as the input size was growing, the instances was taking very long time to get the optimal solution. The reason behind is the WiMAX planning problem has large *integrality gap* property.The *integrality gap* is the maximum ratio between the difference of solution cost of an IP and its LP relaxation to solution cost of an IP, i.e. $Integrality\ Gap = \frac{(Solution\ of\ IP - Solution\ of\ LP\ relaxation\ of\ the\ IP)}{Solution\ of\ IP}$. To see how this gap can be arbitrarily large for the WiMAX planning problem instances we consider the following scenario. We consider a very simplified special case of the WiMAX planning problem. We suppose we have no RS in the network, and there is only one SS with demand 1, and only one BS with capacity $c > 1$ and installation cost f. In the optimal IP solution of this problem the BS is established to serve the SS, and the solution cost is f. In the LP relaxation of the problem,

the SS is also assigned to the same BS, and it is using $\frac{1}{c}$ fraction of the BS capacity, and hence the solution cost for the LP relaxation would be $\frac{f}{c}$, their ratio is $f \div \frac{f}{c} = c$. This c can be arbitrarily large, regardless of the demand of the SS and the establishment cost of the BS. Thus solution of an IP can be arbitrarily larger than its LP relaxation, which can couse a very large integrality gap. CPLEX finds an initial feasible solution (used as the incumbent solution), and the solution to LP relaxation to a problem, and calculates the integrality gap. CPLEX uses branch-and-bound algorithm and applies different cuts to the problem space to improve both the solutions until the integrality gap reduces to 0. Since the WiMAX planning problem has inherent property of large integrality gap, CPLEX takes very long to finish. We, therefore, focus on heuristic approaches to find near optimal solution to the WiMAX planning problem. Subsequent sections present heuristic algorithms that we have experimented with on instances of WiMAX planning problem.

3.2 Greedy Algorithms

The greedy algorithms select SSs, RSs and BSs in different orders, and make SS-RS, SS-BS and SS-BS assignments in such a way that the unsplittable flow, capacity and flow conservation constraints are preserved. While making these assignments the algorithms prefer to make assignment with already open BS or RS rather than opening a new one. At the same time the algorithms prefer to open RS/BS with lower installation cost in some cases, or lower installation cost per unit capacity in the others.

Algorithm 1 presents G-4, one of the 10 greedy algorithms we have designed. The greedy algorithm selects each of the SSs in descending order of their demand and assigns it to an adjacent RS or BS which has sufficient capacity to cover the demand. The algorithm prefers already open RSs or BSs than opening new ones. The algorithm also prefers assigning subscribers to already open RSs than assigning to already open BSs. If no open

RS or open BSs are available adjacent to a SS, only then it chooses to open a RS with the minimum installation cost. When there is no such RS, the algorithm chooses to open a BS with minimum cost.

Once assignment for all SSs are done, we are possibly left with a subset of the RSs carrying traffic from SSs, that are not assigned to any BSs. The open RSs are selected in the descending order of total traffic flow coming to them. These RSs are assigned to one of their adjacent BS with maximum *residual capacity*. Here, *residual capacity* is the capacity of a BS/RS left after satisfying demand of a set of SSs. If no such open base station is available, an adjacent BS with the minimum establishment cost is opened to carry the flow.

This greedy algorithm fails if any SS (or RS) is found whose demand (or flow) cannot be satisfied by any of its adjacent RSs or BSs. Otherwise this greedy algorithm successfully finds a subset of RSs and BSs which can satisfy the bandwidth demand at all the SSs.

Algorithm 1 Greedy G-4

Input: $G = (V, E)$
Output: Solution $A = R' \cup B' : B' \subseteq B, R' \subseteq R$

```
1:  Begin
2:  for each SS in decreasing order of demand do
3:      Assign the SS to one of its adjacent RS/BS chosen according to following order:
4:      1. Open RSs with max residual capacity
5:      2. Open BSs with max residual capacity
6:      3. Closed RSs with min cost
7:      4. Closed BSs with min cost
8:  end for
9:  for each open RS in decreasing order of flow do
10:     Assign the RS to one of its adjacent BS according to following order:
11:     1. Open BSs with max residual capacity
12:     2. Closed BSs with min cost
13: end for
14: End
```

The strategy of the greedy algorithm varies depending on the order of SSs and RSs chosen for assignment, the preference of using already open RSs/BSs vs. opening closed ones. While choosing a RS/BS for assignment to a SS or RS, the choice can be made on different properties such as residual capacity, installation cost, cost per unit-capacity, etc.

In general all the greedy variations we have implemented prefer to satisfy SSs and RSs in decreasing order of demand, and flow respectively. All these algorithms also prefer to use an open BS/RS prior to opening a closed one. The greedy choice properties of different greedy algorithms are summarized in Table 3.1. Algorithm G-3 differs from all others in that it selects SSs in increasing order of demand and assigns RS/BS, while others do the opposite.

The greedy algorithms are all fast, i.e. they can compute solution for input instances with thousands of nodes in milliseconds. The solutions obtained by the greedy algorithms and their running times have been reported in Chapter 5.

3.3 Local Search Algorithms

The local search [1] is a meta heuristic algorithm that can be used on problems which can be modeled as finding a solution from a set of candidate solutions that optimizes some given quality parameters of the solution. The search process projects the solutions into a space (often called the solution space). Local search starts with a candidate solution, and looks for its neighboring solutions to find a better one. As soon a better neighboring solution is found, the search moves its center to the better solution found and keeps moving to its neighbor in the solution space as soon as it gets one better. This process stops when a local optimum is obtained.

The local search algorithms described in this section have graph $G = (V,E)$ as input. The local search starts with an initial solution to the problem A_{init}. All these algorithms produce solution $A_{LocOpt} \subseteq (R \cup B)$. In one step the algorithm searches over its neighbors in the solution space, and moves to a neighboring solution whenever it finds a better one, and resumes the search centering around that solution. The searching process stops when a local minima A_{LocOpt} is reached.

Table 3.1: Greedy choice criterion in different greedy algorithms

	Assignment of SS to RS/BS						Assignment of RS to BS	
	Pref. bet. open RS/BS to use	Order of open RS to consider	Order of open BS to consider	Pref. bet. closed RS/BS to use	Order of closed RS to consider	Order of close BS to consider	Order of open BS to consider	Order of close BS to consider
G-1	RS	min res. cap.	max res. cap.	RS	min install cost per unit cap.	min install cost per unit cap.	min res. cap.	min install cost per unit cap.
G-2	RS	min res. cap.	max res. cap.	RS	min install cost	min install cost	min res. cap.	min install cost
G-3	RS	min res. cap.	max res. cap.	RS	min install cost	min install cost	min res. cap.	min install cost
G-4	RS	max res. cap.	max res. cap.	RS	min install cost	min install cost	max res. cap.	min install cost
G-5	RS	max res. cap.	min res. cap.	RS	min install cost	min install cost	max res. cap.	min install cost
G-6	BS	max res. cap.	max res. cap.	BS	min install cost	min install cost	max res. cap.	min install cost
G-7	any	min res. cap.	min res. cap.	any	min install cost per unit cap.	min install cost per unit cap.	max res. cap.	min install cost
G-8	any	max res. cap.	max res. cap.	any	min install cost per unit cap.	min install cost per unit cap.	max res. cap.	min install cost
G-9	any	max res. cap.	max res. cap.	any	min install cost	min install cost	max res. cap.	min install cost
G-10	any	max res. cap.	max res. cap.	any	min install cost	min install cost	min res. cap.	min install cost

Algorithm 2 Local-Search

Input: An initial solution A_{init}
Output: Solution A_{LocOpt} better or equal to A_{init}

Begin
$A \leftarrow A_{init}$
repeat
    $A_{LocOpt} \leftarrow$ Search-Neighborhood(A,1)
    if A_{LocOpt} is better than A **then**
        $A \leftarrow A_{LocOpt}$
    else
        $A_{LocOpt} \leftarrow$ Search-Neighborhood(A,2)
        if A_{LocOpt} is better than A **then**
            $A \leftarrow A_{LocOpt}$
        end if
    end if
until no update is made over one round of the loop
$A_{LocOpt} \leftarrow A$
return A_{LocOpt}
End

The general framework of the local search algorithms we are proposing is presented in Algorithm 2. Given an initial solution A_{init} the algorithm first searches if there is a feasible solution in any subset of RSs/BSs A_{init} with size $|A_{init}| - 1$, where $|A_{init}|$ is the number of open BS/RS in A_{init}. If no such feasible solution is available, the algorithm closes a RS/BS in the solution and replaces it with a RS/BS with lower installation cost, and checks whether it can be a feasible solution after the replacement. The function Search-Neighborhood is presented in Algorithm 3. The local search moves its center to a better solution as soon as it finds a feasible one, and resumes the search centering that solution. Let A be the candidate solution given to Search-Neighborhood function as input, and let $A' \subseteq A$, $E' \subseteq E, E = (R \cup B) \setminus A$. Local search can have variations depending on how subsets A' and E' are constructed, and how the CheckFeasibility function is implemented.

We have implemented three variations of the local search. The first implementation of local search (LS-1) has $A' = A$ and $E' = E$. CheckFeasibility uses a variation of G-4 presented in Algorithm 1, where the greedy algorithm can only use open RSs/BSs for allocation, thus does not open any closed RSs/BSs. If with a given set (A) of open RS/BS,

the greedy algorithm can make a successful assignment of all the SSs to RSs/BSs and RSs to BSs such that the demands of all SSs are satisfied, A is considered a feasible solution.

Algorithm 3 Search-Neighborhood

Input: Solution A and *neighbourhoodsize*
Output: Best feasible solution A in the neighborhood

```
 1: Begin
 2: for every a ∈ A'(A' ⊆ A) in decreasing order of installation cost do
 3:     Close a from A
 4:     if neighbourhoodsize = 2 then
 5:         for every e ∈ E'(E' ⊆ E, E = ((B ∪ R) \ A)) in increasing order of installation cost do
 6:             if installation cost of e is less than a then
 7:                 Open e in A
 8:                 CheckFeasibility(A)
 9:                 if A is feasible then return A_1
10:                 end if
11:                 Close e in A
12:             end if
13:         end for
14:     end if
15:     if neighbourhoodsize = 1 then
16:         CheckFeasibility(A)
17:         if A is feasible then return A
18:         end if
19:     end if
20:     Open a in A
21: end for
22: return A
23: End
```

The second implementation (LS-2) constructs subset A' by randomly picking elements from A, and does the same to construct E' from E. The CheckFeasibility function here uses the *Max Flow* [30] algorithm to test feasibility. It considers the problem instance with the SSs, open RSs and open BSs as a flow network with SSs as sources and BSs as destinations. It has capacity constraints on RSs and BSs, and flow conservation on the RSs. If the maximum cumulative flow that can be pushed from the sources to destinations through open RSs and BSs is equal to the sum of demands of all the subscribers, CheckFeasibility function returns feasible. This feasibility checking does not guarantee the unsplittable flow property of the problem, i.e. it might happen that the CheckFeasibility function has considered $A \subseteq (B \cup R)$ as feasible which might not be feasible with unsplittable flow. Thus

the solution that LS-2 returns might not be feasible with unspittable flow. A *greedy add procedure* is applied to the outcome of the local search which checks whether the outcome of local search is feasible with unsplittable flow, if not it adds RSs/BSs to this solution until the solution becomes feasible. The *greedy add procedure* works in two steps. In the first step, it considers the flows between SS-RS/BS and RS-BS. Assume SS $s \in S$ sends flows through RSs $R_s \subseteq R$ and BSs $B_s \subseteq B$. The algorithm picks all SS $s \in S$ in an arbitrary order and assigns to the element of $R_s \cup B_s$ which was carrying the most flow from SS s and had enough residual capacity to take the full flow from SS s. If no such RS/BS can be found, SS s remains unsatisfied in this step. The same is done for RS-BS assignments. In the second step the G-4 algorithm is applied to the result from first step to open new RSs/BSs and make assignments for SSs and RSs that were unsatisfied in the first step.

From the experimental results it was observed that LS-1 was taking too long time to run on one of the instance classes we used. The reason was the class of data was very sparse and in such instances trying to replace a RS/BS exhaustively by all closed RSs/BSs was not producing any feasible solutions. The third implementation of local search (LS-3) was designed based on this observation. In this implementation $A' = A$, however, while choosing to add node $e \in E'$ to the solution, it only considers the RSs/BSs which can potentially offset the effect of closing node $a \in A'$ from the solution, and thereby constructs the subset E'. If node a is a RS, and S_a is the set of subscribers adjacent to it, E' contains the set of RSs and BSs adjacent to the elements in S_a. If node a is a BS, S_a and S_r are the set of SSs and RSs adjacent to it respectively, E' contains the RSs and BSs adjacent to S_a and BSs adjacent to S_r. The set E' for each of the RSs and BSs were constructed through a step of pre-computation before the start of the loop in Algorithm 2. The feasibility checking done here in the same way as in LS-1.

All the local searches produce better results than the greedy algorithms. However, the expense of the improvement was the longer running time. The numeric results are presented

in Chapter 5.

3.4 Variable Neighborhood Search Algorithms

Although local search procedure is very effective in searching solutions to different optimization problems, it suffers the problem of getting stuck to a local minima. A Variable Neighborhood Search (VNS) [126] algorithm is mostly based on local search with the difference that whenever local search procedure gets stuck to a local minima, VNS procedure restarts the local search from a different initial solution and keeps track of the best available local minima. This process stops at a defined condition. VNS can be also thought as a multi-start local search process to find the best amongst the local optima. Like all others, these algorithms described have graph $G = (V, E)$ as input, and produces solution $A \subseteq (R \cup B)$. The VNS algorithm is presented in Algorithm 4.

Algorithm 4 VNS

Input: $G(V, E)$
Output: Best local minima A_{best}

Begin
Start with an initial solution A_{init}
$A_{best} \leftarrow A_{init}$
$k \leftarrow k_{init}$
$A \leftarrow A_{init}$
repeat
    $A \leftarrow$ Local-Search(A)
    if A is a better solution than A_{best} **then**
        $A_{best} \leftarrow A$
        $k \leftarrow 1$
    else $k \leftarrow k + 1$
    end if
    $A \leftarrow GenerateKthNeighbor(A, k)$
until there is no update on A_{best} for 3 consecutive rounds of the loop
return A_{best}
End

The VNS procedure described in Algorithm 4 starts with an initial solution A_{init}. This initial solution is generated by the best result from all the greedy procedures described

in Table 3.1. A_{best} keeps track of the best solution produced. In each iteration of the loop the VNS procedure calls the local search described in Algorithm 2. If local search produces a better solution than A_{best}, A_{best} gets updated with the new solution. After each call to local search, A_{init} gets updated by procedure $GenerateKthNeighbor(A_{init}, k)$, here k is the distance between current solution in hand and the k^{th} neighboring solution. $GenerateKthNeighbor(A_{init}, k)$ performs k operations on A_{init} to update it. In each of these k operations $GenerateKthNeighbor(A_{init}, k)$ randomly decides to perform one of three operations: (a) randomly $deleting$ a node from A_{init}, (b) randomly $adding$ a node from $(R \cup B) \setminus A_{init}$ to A_{init}. or (c) $replacing$ a random node from A_{init} by a combination of (a) and (b). The initial value of k is k_{init}. It gets incremented every time the local search fails to get a better solution, and is reset to k_{init} otherwise. k_{max} is the upper bound on k, which for our implementation is set to $|A_{init}|$.

In this implementation of VNS we have used LS-3. The experimental results showed that LS-1 was slow on problem instances generated from a random geometric distribution, which was an incentive to the design of LS-3. LS-3 took a little more time than LS-1 on the randomly generated problem instances because of the pre-computation step, however it was faster on problem instances generated from random geometric distribution.

3.5 Summary

In this chapter we have presented two mathematical formulations of the WiMAX planning problem, and made a comparative study between them. We have presented few heuristics - greedy algorithms, local searches and variable neighborhood search algorithms that can find sub optimal solutions to the problem. In Chapter 5 we present the data specific performance of these algorithms and make a comparative study.

Chapter 4

A Lagrangian Relaxation Based Approach to Solve WiMAX Planning Problem

In sections 3.1.1 and 3.1.2 we have presented two mathematical formulations of the WiMAX planning problem. The first model based on flow conservation and capacity was the most natural way of modeling the problem. It had constants on capacity of BSs and RSs, flow conservation at RSs, and unsplittable flow constraint at the SSs and RSs. The second model was simpler than the first one in terms of types of constraints on those paths. The second model had paths from SSs to BSs, and capacity and unsplittable flow constraints. Although the second was simpler than the first model in terms of constraints, it could not be used for practical situations of large problem instances because of the very large number paths and constraints for each of these paths. In this chapter we present another mathematical formulation of the WiMAX planning problem with simpler constraints than the both models presented in the previous chapter. The IP formulation presented here has the same objective as models presented in Section 3.1. The flow conservation, capacity constraints or unsplittable flow constraints are modeled combinatorially using an equivalent set cover formulation. This model considers a set of trees rooted at the BSs, with SSs as the leaves. The tree can have height of 1 if no RS is present in the tree or 2 if the tree contains RSs and SSs. Such a tree is a *feasible tree* if there is no violation of capacity in any of its RSs or the root BS. These trees are 'maximal' in the sense that no *feasible tree* contain them, while making sure that the capacity constraint at RSs and BSs are not violated. Such trees have establishment costs, which are the sum of the costs of RSs and BS contained in the trees. Given such trees the heuristic algorithms presented here select a sub-collection of the trees so that the total establishment cost is minimized. We present Lagrangian Relaxation [63] based heuristic algorithms to solve the problem. The algorithms presented in

this chapter also provides lower bound to the establishment cost. The problem with this model though is that there can be an exponential number of such trees for a given problem instance, which is difficult to generate and use for computation. We have presented some heuristics to generate trees instead. The experimental results using the algorithm presented here are presented in Chapter 5.

4.1 Modeling the Problem

As in Section 3.1, let $G = (V, E)$ be a tripartite graph with E representing the adjacencies amongst the vertices of G and $V = S \cup R \cup B$, where S is the set of SSs, R is the set of RSs and B is the set of BSs. Let d_s be the demand of SS $s \in S$, f_r and f_b be the installation costs for $r \in R$ and $b \in B$ respectively. Let c_r and c_b be the bandwidth capacity of $r \in R$ and $b \in B$, respectively. Let $\mathcal{T}$ be the set of all possible trees that have the following properties:

1. They are connected subgraphs of G
2. Each of the tree are rooted at one BS $b \in B$
3. The height of a tree is at most 2
4. The node at level 0 is the root, the BS $b \in B$
5. Nodes at level 1 are SSs $s \in S$ or RSs $r \in R$ such that these SSs and RSs are adjacent to BS $b \in B$
6. Nodes at level 2 are only SSs $s \in S$ such that these SSs are adjacent to at least one RSs in the tree
7. The allocations of SSs and RSs to the root BS, and allocations of SSs to RSs satisfy capacity constraints at the RSs and the root BS. Let b be the root of a tree T with set

of RSs R_T and set of SSs S_T. Let S_b be the set of SSs connected to b and S_r be the set of SS connected to RS $r \in R_T$. The tree maintains $c_r \geq \sum_{s \in S_r} d_s$, $\forall r \in R_T$ and $c_b \geq \sum_{s \in S_b} d_s + \sum_{r \in R_T} \sum_{s \in S_r} d_s$. The tree thus also maintains flow conservation property at each RS, since $c_b \geq \sum_{s \in S_b} d_s + \sum_{r \in R_T} \sum_{s \in S_r} d_s$ also implies $c_b \geq \sum_{r \in R_T} \sum_{s \in S_r} d_s$ and therefore any incoming traffic flow into a RS can be forwarded to the root BS of the tree

With set of SSs S_T, set of RSs R_T and the root BS b_T of tree $T \in \mathcal{T}$, the establishment cost of T, f_T is given by:

$$f_T = f_{b_T} + \sum_{r \in R_T} f_r \tag{4.1}$$

where f_{b_T} is the installation cost of the root BS of the tree. We define variables $x_T \in \{0,1\}$ corresponding to each tree $T \in \mathcal{T}$. The problem of satisfying the SSs demands by minimum cost installation of RS and BS can be formulated as follows:

$$Minimize \sum_{T \in \mathcal{T}} f_T x_T \tag{4.2}$$

Subject to:

$$\sum_{T:s \in T} x_T \geq 1, \forall s \in S \tag{4.3}$$

$$\sum_{T:r \in T} x_T \leq 1, \forall r \in R \tag{4.4}$$

$$\sum_{T:b \in T} x_T \leq 1, \forall b \in B \tag{4.5}$$

$$x_T \in \{0,1\}$$

The objective is to minimize the total establishment cost. Constraint (4.3) is a set covering constraint ensuring that each of the SSs are served by one or several RS or BS. Constraint (4.4) is a packing constraint on RSs, enforcing that a RS can get served by at most one BS. Constraint (4.5) is also a packing constraint on BSs ensuring that a BS can be root of at most one tree in the solution tree set. Variable x_T gets the value of 1 when the tree T is selected as a part of the solution, or 0 otherwise. If the value of x_T is 1, root BS and all RSs in the tree are placed, and the subscribers contained in the tree are allocated to the BS and SS. A feasible solution to the problem is a set of such trees that includes all SS the problem instance. The optimal solution is the set of trees with minimum total installation cost.

This formulation of the WiMAX planning problem does not have explicit flow conservation, capacity constraint or unsplittable flow constraint similar to what we have in the formulations in Section 3.1. However, it maintains all these properties of the problem. By their construction mechanism, the trees maintain flow conservation, capacity constraints and unsplittable flow constraints. Constraint (4.5) enforces the fact that a BS in the solution can be the root of at most one of the trees $T \in \mathcal{T}$ (in the solution). Thus constraint (4.5) also ensures that the capacity constraints at BSs are not violated, as long as the trees are feasible. Constraint (4.4), states that a RS can be served by no more than one BS, which enforces the unsplittable flow property into the model. Constraint (4.3) allows the provision for a SS to be covered by multiple trees, however, each of the trees (by their construction) covering a SS in the solution can serve the full demand of the SS by themselves, therefore the subscriber can be allocated to any one of these trees without violating unsplittable flow property at the SSs.

Given all possible trees on a problem instance, the above model can find out the optimal solution for the WiMAX planning problem instance. However, the problem with using this formulation is that the total number of trees maintaining the above properties can be

exponential to the size of the problem. That is, given a problem instance, there can be very large number of trees rooted at different base stations. For very small problem instances, such trees can be generated exhaustively, and given those trees the model can obtain the optimal solution. However, as the problem size increases, the exhaustive process will become computationally intractable.

In this context we decided on generating a set of trees (the tree generation process is described in Section 4.2) using heuristics and using Lagrangian Relaxation [63] based heuristic algorithm to find solutions to the WiMAX planning problem.

4.1.1 Lagrangian Relaxation

Lagrangian relaxation [84, 85] is a method of approximating difficult combinatorial optimization problems in terms of simpler problems. The method penalizes the violations of inequality constraints using non-negative variables for the original problem, these are known as 'Lagrangian multipliers'. In this method, the difficult constraints of the IP are multiplied by corresponding Lagrangian multipliers, and subtracted from the objective function to get to a simpler problem, where the difficult constraints are not present.

As an example of the Lagrangian relaxation process, suppose we want to solve linear programming problem $min\ \{c^T x : Ax \geq b,\ x \in \mathbb{R}^n,\ A \in \mathbb{R}^{m,n}\}$, where the constraints $Ax \geq b$ are complicating. We want to use Lagrangian relaxation to simplify the LP. We introduce the constraint to the objective function and obtain the unconstrained reduced LP: $min\ \{c^T x$ - $\psi^T\ (Ax - b)$: $x \in \mathbb{R}^n,\ A \in \mathbb{R}^{m,n}\}$, where $\psi = \{\psi_1, \psi_2, ..., \psi_m\}$ are non negative Lagrangian multipliers. This form of the LP is called the Lagrangian relxation of $min\ \{c^T x : Ax \geq b,\ x \in \mathbb{R}^n,\ A \in \mathbb{R}^{m,n}\}$. In this case we have taken all the constraints of the original LP to the objective of the Lagrangian relaxation, this is not the case always, sometimes a subset of the constraints are taken to the objective in Lagrangian relaxation. The Lagrangian

multipliers provide similar information about the problem as the LP dual variables does, we can solve the LP relaxation of an IP, compute the dual variables and use them as Lagrangian multipliers.

A very useful property of Lagrangian relaxation is that, if any fixed set of values are given for ψ, the optimal solution to the Lagrangian relaxation will be no larger than the optimal of the original problem. That is, let x' be the optimal solution of original problem and x'' be the optimal solution of the Lagrangian relaxation. We observe that $c^T x' \geq c^T x' - \psi^T(Ax' \geq b) \geq c^T x'' - \psi^T(Ax'' \geq b)$. The first inequality here is true because x' is feasible in the original, and the second is true because x'' is optimal to the LR. Thus the solution to the simpler problem (Lagrangian relaxation) gives a lower bound for the solution to the original (difficult) problem, and gives useful information about the solution to the difficult problem. The above inequality tells us that if we maximize with respect to the dual (model presented in equation (4.9-4.11)), we obtain a tighter lower bound to the objective of the original problem [128]. However, this Lagrangean relaxation bound is not tighter than that of the LP relaxation.

Let π_s, π_r, π_b be non negative dual variables (Lagrangian multipliers) corresponding to each SS $s \in S$, RS $r \in R$ and BS $b \in B$. We multiply the non negative right hand side of constraint (4.3) by π_s, constraint (4.4) by π_r, constraint (4.5) by π_b and subtract it from the objective function (4.2) to obtain the Lagrangian dual objective:

$$\begin{aligned} L(\pi) = Min_{x_T \in \{o,1\}} \{ & \sum_{T \in \mathcal{T}} f_T x_T - \pi_s (\sum_{T: s \in T} x_T - 1) \\ & - \pi_r (1 - \sum_{T: r \in T} x_T) - \pi_b (1 - \sum_{T: b \in T} x_T) \}, \\ & \forall s \in S, \forall r \in R, \forall b \in B \end{aligned} \tag{4.6}$$

Here π is the vector containing $\pi_b \forall b \in B, \pi_r \forall r \in R, \pi_s \forall s \in S$. Thus we relax the model in (4.2-4.5) into an unconstrained model (4.6) using Lagrangian relaxation. Our objective is to find π^* that maximizes $L(\pi)$. Let $f(T,\pi)$ be the cost of establishment of a tree T with respect to dual variables π defined as follows:

$$f(T,\pi)x_T = \sum_{T\in\mathcal{T}} f_T x_T - \pi_s \sum_{T:s\in T} x_T + \pi_r \sum_{T:r\in T} x_T + \pi_b \sum_{T:b\in T} x_T, \forall s \in S, \forall r \in R, \forall b \in B$$

$$\Rightarrow f(T,\pi) = \sum_{r\in T} (\pi_r + f_r) + \sum_{b\in T} (\pi_b + f_b) - \sum_{s\in T} \pi_s \tag{4.7}$$

Using (4.7) we can describe the equation in (4.6) as the following equation (4.8). In the remaining of this chapter we call $f(T,\pi)$ defined in equation (4.7) the 'reduced cost' of T.

$$L(\pi) = Min_{x_T\in\{o,1\}} \sum_{T\in\mathcal{T}} f(T,\pi)x_T - \sum_{r\in R} \pi_r - \sum_{b\in B} \pi_b + \sum_{s\in S} \pi_s \tag{4.8}$$

In the Lagrangian relaxation heuristic procedures we present here, we start with a feasible π. Since $L(\pi)$ is a lower bound on the optimal solution to the original problem, we seek to maximize $L(\pi)$ in an iterative procedure. We calculate the X_T based on the given π (see Section 4.1.2). We use this X_T to calculate π again (see Section 4.1.3). Every $L(\pi)$, given X_T is a candidate for lower bound to the problem, the largest of which is the best lower bound. We continue iterating through these two steps of calculating X_T and π until some stopping condition is met. We use π to reach to a feasible solution to the original problem.

4.1.2 Calculating x_T from π

Assume π_s, π_r, π_b are fixed. From equation (4.7) we see that the $\sum_{s\in S}\pi_s - \sum_{r\in R}\pi_r - \sum_{b\in B}\pi_b$, becomes a constant. Thus to maximize the value of $L(\pi)$, we can compute x_T, $\forall T \in \mathcal{T}$ using the following rules:

1. $x_T = 1$ iff $f(T,\pi) < 0$
2. $x_T = 0$ iff $f(T,\pi) > 0$
3. $x_T \in \{0,1\}$ iff $f(T,\pi) = 0$

This Lagrangian dual objective $L(\pi)$ is a lower bound on the objective of the of the primal model presented in (4.2-4.5) for any selection of $\pi_s \geq 0$, $\pi_r \geq 0$ and $\pi_b \geq 0$ [141]. This is because, if x_T is feasible then the constraint violations cannot be positive.

4.1.3 Calculating π from x_T

The dual of the formulation in (4.2-4.5) is as follows:

$$Maximize \sum_{s\in S}\phi_s - \sum_{r\in R}\phi_r - \sum_{b\in B}\phi_b \tag{4.9}$$

Subject to:

$$\sum_{s\in S_T}\phi_s - \sum_{r\in R_T}\phi_r - \phi_{b_T} \leq f_T, \quad \forall T \in \mathcal{T} \tag{4.10}$$

$$\phi_s, \phi_s, \phi_s \geq 0 \tag{4.11}$$

We have already described a process to assign values for X_T when values for π_s, π_r, π_b are fixed. Given X_T fixed, we intend to calculate π_s, π_r, π_b. To obtain π_s, π_r, π_b, we

can solve the formulation in equation (4.9-4.11) and use the values of ϕ_s, ϕ_s, ϕ_s for π_s, π_r, π_b respectively. The ϕ_s, ϕ_r, ϕ_b thus obtained are the optimal values to maximize objective in equation (4.9). Since the model in equation (4.9-4.11) is the dual of model in equation (4.2-4.5), optimal ϕ_s, ϕ_r, ϕ_b values for model in equation (4.9-4.11) are the optimal dual variables for model in equation (4.2-4.5). The formulation in equation (4.6) is relaxation of model in (4.2-4.5), therefore using ϕ_s, ϕ_r, ϕ_b obtained by solving model in equation (4.9-4.11) as π_s, π_r, π_b, given X_T fixed, maximizes the lower bound $L(u)$ in (4.6).

Computing an optimal multiplier vector by solving an LP is generally time consuming for large problem instances. This is not suitable to be used inside an iterative process. We have used an approach to find near optimal Lagrangian multipliers similar to the one used in [32]. Caprara et. al. in [32] presented a heuristic approach that used Lagrangian relaxation to approximate set covering problem. In this approach Caprara et. al. presented an iterative procedure that starts with an arbitrary set of Lagrangian multipliers, calculates incumbent solution (primal variables) based on the multipliers. They also proposed a heuristic that allows to compute near optimal Lagrangian multipliers. This process is repeated until the the incumbent solution cannot be further improved. We have used a similar approach based on the proposal of Caprara et. al. [32]. Given these x_T and values for π_s, π_r, π_b, the approach used by Caprara et. al. uses subgradient vector $g(\pi) \in R^{|V|}$ associated with a given π. The idea of using subgradient vector follows from equation (4.7) and (4.8). Given fixed X_T, from equation (4.7) and (4.8) we have:

$$
\begin{aligned}
L(\pi) &= \sum_{T\in\mathcal{T}} (\sum_{r\in T}(\pi_r + f_r) + \sum_{b\in T}(\pi_b + f_b) - \sum_{s\in T}\pi_s)x_T - \sum_{r\in R}\pi_r - \sum_{b\in B}\pi_b + \sum_{s\in S}\pi_s \\
&= \sum_{T\in\mathcal{T}} f_T x_T \\
&+ \sum_{b\in B}(\sum_{T:b\in T} x_T - 1)\pi_b \\
&+ \sum_{r\in R}(\sum_{T:r\in T} x_T - 1)\pi_r \\
&+ \sum_{s\in S}(1 - \sum_{T:s\in T} x_T)\pi_s
\end{aligned} \tag{4.12}
$$

In equation (4.12) we intend to maximize $L(\pi)$. The $\sum_{T\in\mathcal{T}} f_T x_T$ produces a constant in this equation, given fixed X_T. The terms $(\sum_{T:b\in T} x_T - 1)$, $\sum_{r\in R}(\sum_{T:r\in T} x_T - 1)$ and $\sum_{s\in S}(1 - \sum_{T:s\in T} x_T)$ also generates positive or negative constants, given fixed X_T. The positive constants from $(\sum_{T:b\in T} x_T - 1)$, $\sum_{r\in R}(\sum_{T:r\in T} x_T - 1)$ and $\sum_{s\in S}(1 - \sum_{T:s\in T} x_T)$ positively contribute to the $L(\pi)$, while negative terms contribute to the $L(\pi)$ negatively. If we adjust the elements of π vector by increasing the entries corresponding to the positive constants and and decreasing the ones corresponding to the negative constants, we get higher value of $L(\pi)$. Thus the terms $(\sum_{T:b\in T} x_T - 1)$, $\sum_{r\in R}(\sum_{T:r\in T} x_T - 1)$ and $\sum_{s\in S}(1 - \sum_{T:s\in T} x_T)$ can be used as a direction of improvement for the π, as used in case of [32]. We define vector $g(\pi)$ as follows:

$$
g_s(\pi) = 1 - \sum_{T:s\in T} x_T, \forall s \in S \tag{4.13}
$$

$$
g_r(\pi) = \sum_{T:r\in T} x_T - 1, \forall r \in R \tag{4.14}
$$

$$
g_b(\pi) = \sum_{T:b\in T} x_T - 1, \forall b \in B \tag{4.15}
$$

These subgradients $g(\pi)$ are then normalized. For normalization we multiplied $g(\pi)$ by the difference between upper bound and lower bound, and divided the product by $||g(\pi)||^2$, where $||g(\pi)||^2 = \sum(g(\pi))^2$, see equation (4.16). This is same as what has been done in [32].

Our iterative procedure starts with an allocation of positive values to the Lagrangian multiplier vector π using solution of G-4 (Algorithm 1) presented in Section 3.2, and generates a sequence of non negative Lagrangian multiplier vectors π^0, π^1, π^2, ..., where π^0 is the initial allocation. Given π^k and its corresponding values for vector $X^k(x_T), \forall T \in \mathcal{T}$, π^{k+1} is computed using the following formula [32]:

$$\pi_i^{k+1} = Max\{\pi_i^k + \lambda \frac{UB - L(u^k)}{||g(\pi^k)||^2} g(\pi^k),\ 0\}, for\ i \in (S \cup R \cup B) \tag{4.16}$$

In equation (4.16) UB is the upper bound on the WiMAX planning problem and $\lambda > 0$ is the step size. The UB is set to the best solution found. Initially it is computed by the greedy heuristic G-4. λ is initially set to 0.1. It is updated after every $p = 10$ iterations of subgradient generation. If the difference between the best and worst lower bounds is more than 1%, the current value of λ is halved. If the difference is less than 0.1%, the current λ is multiplied by 1.5 [32]. Based on equation (4.16) and rules to update the step size first variation of Lagrangian relaxation algorithm (LR-1 presented in Algorithm 8) has been implemented.

4.2 Tree Generation

The model in (4.2-4.5) uses a set of trees $\mathcal{T}$. We have already discussed in Section 4.1 this set $\mathcal{T}$ can have very large number of trees for the large instances in the best case,

and why it is not feasible in terms of running time and memory to generate all of such trees to construct set $\mathcal{T}$. We construct $\mathcal{T}$ in a two step process. The first step uses greedy approaches to generate the initial set of trees. At the end of each step we ensure that $\mathcal{T}$ does not contain multiple copies of the same tree. The second step uses dual information to generate more trees. In the following two subsections we discuss these two steps.

4.2.1 Greedy Tree Generation

Algorithm 5 Tree Generation

Input: $G = (V,E), V = S \cup R \cup B$
Output: Solution $\mathcal{T} = \{T : T = b_T \cup S_T \cup R_T\}, b_T \in B, R_T \subseteq R, S_T \subseteq S$

Begin

1. Order all BS based on their capacity, cost, capacity-cost ratio, or in random order (both ascending and descending) to construct the following set of ordered lists: B_{order} = $\{B_{(cap,asc)}, B_{(cap,desc)}, B_{(cost,asc)}, B_{(cost,desc)}, B_{(cost-cap.ratio,asc)}, B_{(cost-cap.ratio,desc)}, B_{(rand)}\}$
2. Order all RS based on their capacity, cost, capacity-cost ratio, or in random order (both ascending and descending) to construct the following set of ordered lists: R_{order} = $\{R_{(cap,asc)}, R_{(cap,desc)}, R_{(cost,asc)}, R_{(cost,desc)}, R_{(cost-cap.ratio,asc)}, R_{(cost-cap.ratio,desc)}, R_{(rand)}\}$
3. Order all SS based on their demand, number of RS/BS adjacent to it (adjacency number), ration between demand and adjacency number, or in random order (both ascending and descending) to construct the following set of ordered lists: S_{order} = $\{S_{(dem,asc)}, S_{(dem,desc)}, S_{(adj,asc)}, S_{(adj,desc)}, S_{(dem-adj.ratio,asc)}, S_{(dem-adj.ratio,desc)}, S_{(rand)}\}$

for $o_B \in B_{order}$ **do**
  for $o_R \in R_{order}$ **do**
    for $o_S \in S_{order}$ **do**
      TreeGeneration-1 (G, o_B, o_R, o_S)
    end for
  end for
end for

End

In this step we used all the 10 versions of greedy algorithm presented in Section 3.2 to construct set $\mathcal{T}$ first. Apart from that, we have also used the procedure presented in the Algorithm 5 and 6 to generate more trees. The algorithm judges BSs, RSs and SSs in different orders. For BSs and RSs, the ordering be can be done by installation cost, capacity of the RS/BS, the ratio between installation cost and capacity for BR/RS, or random ordering.

Algorithm 6 TreeGeneration-1

Input: $G = (V,E)$, $V = S \cup R \cup B$, BS ordering o_B, RS ordering o_R, SS ordering o_R
Output: Solution $\mathcal{T} = \{T : T = b_T \cup S_T \cup R_T\}$, $b_T \in B, R_T \subseteq R, S_T \subseteq S$

1: **Begin**
2: **for** all BS $b \in o_B$ **do**
3: **for** all RS $r \in o_R$ **do**
4: **if** $(r,b) \in E$ and b has enough residual capacity to cover the capacity of r **then**
5: Assign r to b
6: **for** all SS $s \in o_S$ **do**
7: **if** $(s,r) \in E$, s has not already been assigned to any RS and r has enough residual capacity to cover the demand of s **then**
8: Assign s to r
9: **end if**
10: **end for**
11: **end if**
12: **end for**
13: **for** all SS $s \in o_S$ **do**
14: **if** $(s,b) \in E$, s has not already been assigned to any RS or BS, and b has enough residual capacity to cover the demand of s **then**
15: Assign s to b
16: **end if**
17: **end for**
18: add this tree generated in steps$(2-17)$ to $\mathcal{T}$
19: **end for**
20: **End**

Given any of these ordering, this algorithm picks a BS, adds RS in accordance with the order of RS given until the BS capacity is saturated, or the list of RSs adjacent to a BS is saturated. Once done with allocating a RS, the algorithm add SSs (which has not already been allocated to a RS) to the RS in accordance with the ordering of SSs, than considers the next RS. After this, if the BS has more capacity left and some SSs exist (which has not been allocated to any RS) adjacent to the BS the remaining SSs are allocated to the BS in given order of SSs. After this step we have a tree, we add it to $\mathcal{T}$, if it is not already existing in the set, and proceed with the next BS in the given order.

With the list of trees $\mathcal{T}$ we obtain from the previous steps, we perform another step of tree generation to increase the initial set of trees. In this step, we consider each tree $T \in \mathcal{T}(T = S_T \cup R_T \cup b_T)$, we take off each RS $r \in R_T$ and set of all SSs $S'(s \in S_T \, and \, (s,r) \in E)$ from T. We try to replace r by another RS $r_1 \in (R \setminus R_T)$ and add to this r_1 as many SSs

$s_1 \in (S \setminus (S_T \setminus S'))$ as possible without violating the capacity. Thus we generate a set of new trees $\mathcal{T}_1$. We add the trees in $\mathcal{T}_1 \setminus \mathcal{T}$ to $\mathcal{T}$.

4.2.2 Tree Generation Using Dual information

From Section 4.2.1 we have a set of initial trees in the set $\mathcal{T}$. We want to populate the set $\mathcal{T}$ with more trees. We have discussed in Section 4.1.3 how the initial values of the Lagrangian multipliers (π) can be calculated with the set of initial trees. This requires to solve the LP relaxation of the set cover formulation and to use the optimal dual variables. Rather than solving an LP to get the initial set of Lagrangian multipliers, heuristic algorithms are presented in later part of this chapter that enables us calculate the initial set of π using Lagrangian relaxation process (see detail in Section 4.3). These values of π are the entries corresponding to different BSs, RSs and SSs.

For convenience of explanation, we use matrix representation of the set of trees, π-variables, tree variables and tree costs to express the IP in (4.2-4.5). Let N be the matrix containing the trees $T \in \mathcal{T}$ as columns. The number of rows in the matrix is equal to the number of vertices in G, each row corresponds to an element of $V = S \cup R \cup B$. The column of N corresponding to tree $T \in \mathcal{T}$ has an entry of 1 corresponding to all the SS $s \in S_T$ and an entry of 0 for all $s \notin S_T$, a -1 entry corresponding to every RS and the root BS contained by T, and entry of 0 for rest of the BSs and RSs. Let C_N be a column vector representing the cost corresponding to columns in N, X_N be the column vector representing variables corresponding to the trees in N, and b be the right hand side column vector of (4.2-4.5). Let π, be a column vector representing the dual variables.

We convert this IP to a LP 'standard form' [43] where the constraint matrix $A = [B|N]$. Here B is an identity matrix corresponding to the initial basic feasible solution. Let X_B and C_B be the column vectors of variables and costs corresponding to columns in B. The model

in (4.2-4.5) can be written as follows:

$$Minimize\ Z = C_B^T X_B + C_N^T X_N \tag{4.17}$$

Subject to:

$$BX_B + NX_N = b \tag{4.18}$$

From (4.18) we have:

$$X_B = B^{-1}b - B^{-1}NX_N \tag{4.19}$$

From (4.17) and (4.19) we have:

$$\begin{aligned} Z &= C_B^T B^{-1}b - C_B^T B^{-1}NX_N + C_N^T X_N \\ &= \pi^T b - \pi^T NX_N + C_N^T X_N, \quad \pi^T = C_B^T B^{-1} \\ &= \pi^T b + (C_N^T - \pi^T N)X_N \end{aligned} \tag{4.20}$$

The equation (4.20) is the matrix representation of formulation in (4.6) and (4.8). We are supposed to minimize Z. This set of trees in $\mathcal{T}$ are the columns in N. The column generation process can be defined as adding a column vector N_1 to the matrix N. Given π, with respect to equation (4.20), we have $\pi^T b$ constant, therefore the new column should be generated in a way such that $(C_N^T - \pi^T N_1)$ is negative. This means:

$$-\sum_{b\in B}\pi_b x_b-\sum_{r\in R}\pi_r x_r+\sum_{s\in S}\pi_s x_s\geq\sum_{b\in B}f_b x_b+\sum_{r\in R}f_r x_r,\quad \forall x\in 0,1$$

$$\Rightarrow\sum_{b\in B}(f_b+\pi_b)x_b+\sum_{r\in R}(f_r+\pi_r)x_r-\sum_{s\in S}\pi_s x_s\leq 0,\quad \forall x\in 0,1 \tag{4.21}$$

From equation (4.21) we see that a weight of $(f_b+\pi_b)$ associated with every BS b, a weight of $(f_r+\pi_r)$ associated with every RS r, and a weight of $-\pi_s$ associated with every SSs. In Algorithm 7 we present a greedy heuristic procedure that generates trees intending to minimize the sum of these weights. This procedure is run iteratively with new set of π_b, π_r, π_s calculated over the entire set of trees $\mathcal{T}$ generated until that point at every iteration. This procedure is iterated to generate a number of trees until we cannot generate any more trees using the mentioned weights. This procedure is called column generation in the literature. More details on column generation can be found in [50].

4.3 Applying Lagrangian multipliers to Solve WiMAX Planning Problem

In this section we propose two 3-phase heuristic procedures to improve the values for Lagrangian multipliers to obtain better approximation to the WiMAX planning problem. The first implementation of Lagrangian relaxation (LR-1) is presented in Algorithm 8.

In the first phase of the 3-phase heuristic LR-1 we do the necessary initialization. The second phase is an iterative process to generate a sequence of Lagrangian multiplier vectors π^k and corresponding vector X^k tree variables, until some stopping condition is met. In each step of this phase the best set of Lagrangian multipliers π^* (that corresponds to the best lower bound found so far) is kept. The third and the final phase uses π^* obtained from

Algorithm 7 Column Generation

Input: $G = (V,E)$, $V = S \cup R \cup B$, $(f_b + \pi_b) \forall b \in B$, $(f_r + \pi_r) \forall r \in R$, $\pi_s \forall s \in S$
Output: Solution $\mathcal{T} = \{T : T = b_T \cup S_T \cup R_T\}$, $b_T \in B, R_T \subseteq R, S_T \subseteq S$

Begin
for all BS $b \in B$ **do**
  for all SS $s \in S$ in decreasing order of π_s **do**
    if $(s,b) \in E$ and b has enough residual capacity to cover s **then**
      Assign s to b
    end if
  end for
  for all RS $r \in R$ in increasing order of $(f_r + \pi_r)$ **do**
    $f\pi_r \leftarrow (f_r + \pi_r)$
    for all SS $s \in S$ which has not already been assigned to any open RS/BS so far in decreasing order of π_s **do**
      if $(s,r) \in E$ and r has enough residual capacity to cover s **then**
        Assign s to r
        $f\pi_r \leftarrow f\pi_r - \pi_s$
      end if
    end for
  end for
  for all RS $r \in R$ in increasing order of $f\pi_r$ **do**
    if $(b,r) \in E$ and b has enough residual capacity to cover the flow in r **then**
      Assign s to r
    end if
  end for
  Add the tree rooted in b generated in this loop to $\mathcal{T}'$
end for
for all BS $b \in B$ **do**
  for all RS $r \in R$ in increasing order of $(f_r + \pi_r)$ **do**
    if $(r,b) \in E$ and b has enough residual capacity to cover flow equal to the capacity of r **then**
      Assign r to b
      for all SS $s \in S$ which has not already been assigned to any RS in decreasing order of π_s **do**
        if $(s,r) \in E$ and r has enough residual capacity to cover s **then**
          Assign s to r
        end if
      end for
    end if
  end for
  for all SS $s \in S$ which has not already been assigned to any RS/BS in decreasing order of π_s **do**
    if $(s,b) \in E$ and b has enough residual capacity to cover s **then**
      Assign s to b
    end if
  end for
  Add the tree rooted in b generated in this loop to $\mathcal{T}'$
end for
Add the trees in $\mathcal{T}'$ to $\mathcal{T}$
End

the second phase to construct a solution to the WiMAX planning problem. Algorithm 8 gives a high level description of the heuristic.

In the iterative phase of LR-1, first we compute vector X, using rules described in Section 4.1.2 and reduced cost of the trees $f(T,\pi)$. After that we use π and X to compute the new upper bound and lower bound. If this new upper bound is better than the best one known so far, we update the best known upper bound with the newly computed one. If the newly computed lower bound is better than the best lower bound known so far, we copy the vector π in π^{old} (vector corresponding to the best known lower bound), update the best known lower bound and the π vectors with the new one, and compute π^{i+1} using formula (4.16). In every 10th step of this iterative phase we update step size λ using the same rules in [32]. If the newly computed upper bound is not better than the best one known so far, the step size λ is halved, and vector π is set to π^{old}. The iterative process terminates when the improvement of lower bound over 100 steps is less than 1% or we reach a number of iterations equal to $10 \times number\ of\ nodes\ in\ the\ instance$. This stopping condition is also based on the proposal in [32]. After this phase terminates, we have obtained a lower bound, and its corresponding π-vector, we name it π^*. We use this π^* in the third phase to greedily obtain a feasible solution to the problem instance.

Algorithm 9 presents the third phase of the heuristic procedure in Algorithm 8. In this phase of constructing the final solution, the reduced cost $f(T,\pi)$ (see equation 4.8) is calculated for all the trees in $\mathcal{T}$ using π^*. The trees are sorted in ascending order of their reduced cost. After this we enter into an iterative process with the list of trees in above mentioned order. We pick the first tree, and add the BS and RS of this tree to the solution *Sol*, then we check whether *Sol* can satisfy all the demands of SS in the network using a method called $CheckFeasibility()$ (see Section 3.3 on LS-1, LS-3 for detail), if *Sol* is feasible we return the solution, otherwise we enter into the next iteration of the loop with the next tree in order. The iteration stops when we have exhausted all trees in the ordered

Algorithm 8 3-Phase heuristic algorithm LR-1

Input: $G = (V, E)$, $V = S \cup R \cup B$, Set of trees $\mathcal{T}$
Output: Solution *Sol*, Lower bound $L(\pi)$

Begin
Phase 1: Initialization
$\lambda \leftarrow 0.1$
Upper bound UB $\leftarrow$ the cost of solution obtained by G4 (Algorithm 1)
Initialize Lagrangian multiplier vector π^0 using G4 (Algorithm 1) such that the reduced cost of the trees from G4 are negative, and reduced cost of all other trees become 0 or positive.
Initialize $L(\pi^0)$ using π^0
$i \leftarrow 0$
Phase 2: Iterative ascent procedure
repeat
    Compute X^i, given π^i using rules described in Section 4.1.2
    Compute new upper bound UB_{new}, given X^i, π^i using $ConstructSolution(G, \pi^i)$
    Compute lower bound $L(\pi^i)_{new}$, given X^i and π^i using equation (4.6)
    if $UB_{new} < UB$ **then**
        $UB \leftarrow UB_{new}$
    end if
    if $L(\pi^i)_{new} > L(\pi^i)$ **then**
        $\pi^{old} \leftarrow \pi^i$
        $L(\pi^i) \leftarrow L(\pi^i)_{new}$
        Compute π^{i+1} using formula (4.16)
        if $(i\%10 = 0)$ **then**
            Update λ
        end if
    else
        $\pi^{i+1} \leftarrow \pi^{old}$
        $\lambda \leftarrow \lambda/2$
    end if
    $i \leftarrow i + 1$
until (Improvement of *LB* over 100 iterations is less than 1%) or (this loop iterates more than $10 \times$ *Number of nodes in the instance* times)
after this loop we have π^* and that will be used in the next phase, and lower bound $L(\pi^*)$
Phase 3: Solution construction
$Sol \leftarrow ConstructSolution(G, \pi^*)$
End

list who have non-positive reduced cost. After this we apply an *add procedure* (described in Section 3.3 for LS-2) on *Sol* obtained from the iterative step to make it a feasible solution, and return it.

Algorithm 9 ConstructSolution

Input: $G = (V,E)$, $V = S \cup R \cup B$, Lagrangian multipliers π^*, Set of trees $\mathcal{T}$
Output: Solution *Sol*

```
1:  Begin
2:  Calculate reduced cost for all there T ∈ 𝒯 using equation 4.8 and store the result in TreeRedCost
3:  Sort trees using ascending order of TreeRedCost
4:  for tree T ∈ 𝒯 in ascending order of TreeRedCost with reduced cost of T not positive do
5:      Add the BS and RSs of T to Sol
6:      if CheckFeasibility(Sol) then
7:          return Sol
8:      end if
9:  end for
10: Apply add procedure (described in Section 3.3 for LS-2) on Sol to make it a feasible solution
11: return Sol
12: End
```

Although the number of iterations of LR-1 was never larger than ten times the number of nodes in each problem instance as observed by [32], the improvement of the lower bound for LR-1 was slow. This led us to implement the second variation of Lagrangian relaxation LR-2. LR-2 is also a 3 phase procedure. First phase of LR-2 has a subset of operations as in LR-1. The 3rd phase is the same. The iterative phase (2nd phase) is different. The basic difference between LR-1 and LR-2 is in the implementation of normalizing the subgradient vectors, in the updating of the Lagrangian multipliers, and in the updating of step sizes. LR-2 (as shown in our experimental experience presented in Chapter 5) is a faster process obtaining better lower bounds in all instances. The main credit of this runtime improvement goes to the way of updating step size used in LR-2. The LR-2 also does not calculate or use any upper bound information, which also contributed to the run time enhancement. Algorithm 10 presents a high level description of LR-2 heuristics.

The first phase of LR-2 heuristics initializes the step size λ to 0.01, initialize Lagrangian multiplier vector π^0 using information obtained from G-4, and calculates initial lower

Algorithm 10 3-Phase heuristic algorithm LR-2

Input: $G = (V, E)$, $V = S \cup R \cup B$, Set of trees $\mathcal{T}$
Output: Solution Sol, Lower bound $L(\pi)$

Begin
Phase 1: Initialization
$\lambda \leftarrow 0.01$
Initialize Lagrangian multiplier vector π^0 using G-4 (Algorithm 1) such that the reduced cost of the trees from G4 are negative, and reduced cost of all other trees become 0 or positive.
Initialize $L(\pi^0)$ using π^0
$i \leftarrow 0$
Phase 2: Iterative ascent procedure
repeat
    if $i \neq 0$ **then** $\lambda \leftarrow \lambda/2$
    end if
    Compute X^i, given π^i using rules described in Section 4.1.2
    Compute π^i_a using π^i, X^i and step size λ using equation (4.22)
    Compute X^i_a, given π^i_a using rules described in Section 4.1.2
    Compute lower bound aL, given X^i_a and π^i_a using equation (4.6)
    Compute π^i_b using π^i, X^i and step size $2 \times \lambda$ using equation (4.22)
    Compute X^i_b, given π^i_b using rules described in Section 4.1.2
    Compute lower bound bL, given X^i_b and π^i_b using equation (4.6)
    if $aL \leq bL$ **then**
        while $aL \leq bL$ **do**
            $\lambda \leftarrow 2 \times \lambda$
            $aL \leftarrow bL$
            $\pi^i_a \leftarrow \pi^i_b$
            Compute π^i_b using π^i, X^i and step size $2 \times \lambda$ using equation (4.22)
            Compute X^i_b, given π^i_b using rules described in Section 4.1.2
            Compute lower bound bL, given X^i_b and π^i_b using equation (4.6)
        end while
    else
        while $aL > bL$ **do**
            $\lambda \leftarrow \lambda/2$
            $bL \leftarrow aL$
            $\pi^i_b \leftarrow \pi^i_a$
            Compute π^i_a using π^i, X^i and step size λ using equation (4.22)
            Compute X^i_a, given π^i_a using rules described in Section 4.1.2
            Compute lower bound aL, given X^i_a and π^i_a using equation (4.6)
        end while
    end if
    $L(\pi^i) \leftarrow bL$
    $pi^i \leftarrow \pi^i_b$
until (Improvement of LB over 100 iterations is less than 0.1%) or (this loop iterates more than $10 \times$ *Number of nodes in the instance* times)
after this loop we have π^* and that will be used in the next phase, and lower bound $L(\pi^*)$
Phase 3: Solution construction
$Sol \leftarrow ConstructSolution(G, \pi^*)$
End

bound $L(\pi^0)$ using π^0. Given π and step size λ, we calculate two next lower bounds aL and bL, which are in distance λ and $2 \times \lambda$ from the current $L(\pi^0)$. If $bL > aL$, we keep doubling step size λ until we find a pair of lower bounds aL, bL with $aL > bL$. Otherwise we keep halving step size λ until we have $bL \geq aL$ again. Essentially, we use binary search to get closer to the optimum step size. After this we update $L(\pi)$ and its corresponding Lagrangian multiplier vector π with the best lower bound bL and its corresponding π_b found in this loop. We iterate through this process until we reach the same stopping condition as in LR-1 algorithm. The phase 3 for LR-2 is same as that of LR-1.

Given π^k and its corresponding values for vector $X^k(x_T), \forall T \in \mathcal{T}$, the updated Lagrangian multiplier vectors π^{k+1} is in case of LR-2 computed using the following formula:

$$\pi_i^{k+1} = Max\{\pi_i^k + \lambda \frac{g(\pi^k)}{||g(\pi^k)||},\ 0\}, for\ i \in (S \cup R \cup B) \tag{4.22}$$

In equation (4.22) λ is the step size. π_i^k is the current Lagrangian multiplier vector, π_i^{k+1} is the updated Lagrangian multiplier vector $g(\pi^k)$ is the direction of improvement (see equation 4.13-4.15). This way of updating set of Lagrangian multipliers differs from LR-1 in that in this case we multiply the step size λ with the normalized direction of improvement $\frac{g(\pi^k)}{||g(\pi^k)||}$, where as in the previous case we used to multiply λ with the normalized direction of improvement and the difference between upper and lower bounds. The difference of normalizing $g(\pi^k)$ in case of LR-1 and LR-2 can be clearly observed from equation (4.16) and (4.22). This heuristic was faster and obtained better lower bounds than LR-1.

The phase 1-2 of Algorithm 8 and 10 is intended to estimate some large feasible values for Lagrangian multipliers π. Then in the phase 3 of these algorithms we construct a feasible solution using these π values. The largest values of for π can be obtained by solving the model in (4.9-4.11) and use the optimal duals as π. In algorithm LR-3, we have replaced

the phase 1-2 of Algorithm 8 (i.e. lines 2 29) with a call to ILOG CPLEX 12.1.0 [92] to solve the model in (4.9-4.11) and used the optimal duals as Lagrangian multipliers as π for phase 3 to obtain a feasible solution to the problem. This algorithm gives us $L(\pi)$ obtained from optimal dual variables, and a solution constructed from that π.

4.4 Summary

In this chapter we have described two Lagrangian relaxation based heuristic algorithms to approximate WiMAX planning problem. The test results and analysis on the data specific cases are presented in Chapter 5.

Chapter 5

Experimental Results

In this chapter we present the experimental evaluations of the algorithms presented in Chapter 3 and Chapter 4. In Section 5.1 we discuss the experimental setup and data used for the experiments. Section 5.2 presents the solutions obtained by CPLEX, sections 5.3-5.5 present the experiment results obtained by our greedy, local search and VNS algorithms respectively. Section 5.6 presents results obtained from algorithms presented in Chapter 4. Finally Section 5.7 summarizes all the results.

5.1 Experimental Setup and Data

All the algorithms presented in Chapter 3 and 4 were implemented using C++ programming language. The experimental results presented in this chapter were generated using a computer with AMD Athlon *ii* X4 630 processor and 8 GB of RAM.

We have reported test results on 19 problem instances of 500-5000 nodes. These instances have been generated over a varying distribution of sparsities, installation costs, demands of SSs, capacities of RSs/BSs, etc. Instances $ta1-ta5$ and $tb1-tb5$ are randomly generated tripartite graphs with vertex set comprising of SSs, RSs and BSs, and edge set comprising of SS-RS, SS-BS and RS-BS connectivities. Instances $tc1-tc5$ and $te1-te4$ are generated from a geometric distribution of SSs, RSs and BSs on a two dimensional plane. Transmission ranges are defined for BS and RS. BS-SS and BS-RS links have been established when SSs/RSs were within the transmission range of a BS. SS-RS links have been established between SSs and RSs, when the SSs were within transmission range of RSs, and BS-RS links have been established between RSs and BSs, when the RSs were within the transmission range of BSs. The demands of SSs, installation costs of RSs and

Table 5.1: Problem instances

Instance Type	Name of Instances	Number of SSs / RSs / BSs	Total number of Nodes	Number of Edges	Average Degree
Random Graphs	ta1	300 / 125 / 75	500	14031	28.06
	ta2	600 / 250 / 150	1000	55502	55.50
	ta3	1200 / 500 / 300	2000	220914	110.46
	ta4	2400 / 1000 / 600	4000	888378	222.09
	ta5	3000 / 1250 / 750	5000	1387895	277.58
	tb1	450 / 187 / 113	750	31358	41.81
	tb2	750 / 313 / 187	1250	86257	69.01
	tb3	1500 / 626 / 374	2500	346598	138.64
	tb4	2400 / 1000 / 600	4000	888133	222.03
	tb5	3000 / 1250 / 750	5000	1389378	277.88
Geometric Graphs	tc1	300 / 125 / 75	500	540	1.08
	tc2	600 / 250 / 150	1000	1125	1.13
	tc3	1200 / 500 / 300	2000	2162	1.08
	tc4	2400 / 1000 / 600	4000	4415	1.10
	tc5	3000 / 1250 / 750	5000	5842	1.17
	te1	300 / 125 / 75	500	641	1.28
	te2	600 / 250 / 150	1000	1919	1.92
	te3	1200 / 500 / 300	2000	6247	3.12
	te4	2400 / 1000 / 600	4000	11491	2.87

BSs were randomly generated from natural distribution, capacities of RSs and BSs were randomly generated from a natural distribution for both of the instance classes. The number of SS, RS and BS, the number of edges, and average degree of nodes in each of the problem instances are presented in Table 5.1. From the average degree we observe that the geometric instances are much sparser than randomly generated instances.

5.2 ILOG CPLEX

In Table 5.2 we report computational results by using ILOG CPLEX 12.1.0 [92]. All the instances were ran on CPLEX using a time limit of 180 minutes. Instances $ta1 - ta5$ and

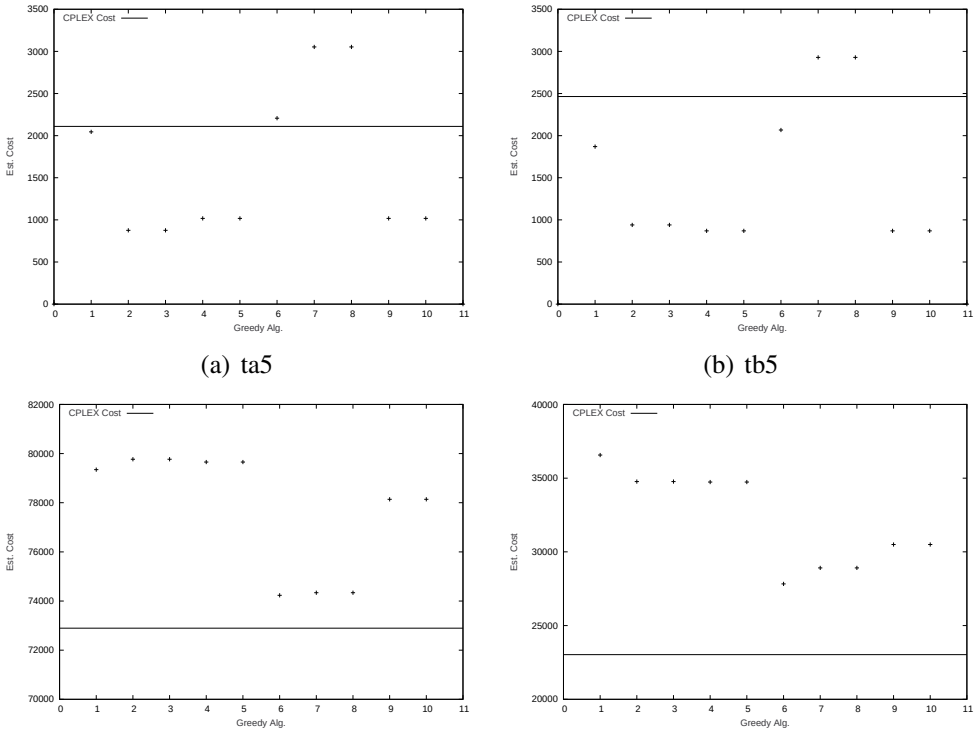
Table 5.2: Results from CPLEX

Instance Type	Problem Instance	Establishment Costs	Integrality Gap	Running Time (Min.)
Random Graphs	ta1	447	94.05%	180
	ta2	1327	94.18%	180
	ta3	1624	96.60%	180
	ta4	1854	99.92%	180
	ta5	2110	99.94%	180
	tb1	501	98.88%	180
	tb2	1372	98.30%	180
	tb3	1613	97.35%	180
	tb4	2029	96.90%	180
	tb5	2464	99.91%	180
Geometric Graphs	tc1	4610	0.00%	≤ 0.002
	tc2	10724	0.00%	≤ 0.002
	tc3	25678	0.06%	≤ 0.002
	tc4	56464	0.00%	≤ 0.002
	tc5	72895	0.00%	≤ 0.002
	te1	3618	0.00%	≤ 0.002
	te2	6587	0.04%	≤ 0.002
	te3	9564	0.27%	0.02
	te4	23028	0.00%	0.05

$tb1 - tb5$ reached 180 minutes and yet had integrality gap around $94\% - 99\%$. Instances $tc1 - tc5$ and $te1 - te4$ finished running and optimal results for them have been reported.

5.3 Greedy Algorithms

Figure 5.1 presents the performance of greedy algorithms on 4 large instances. The horizontal line represents the value solution obtained by CPLEX in at most 3 hours. The points represent the establishment cost that different greedy algorithms scored. The lower the values in the y-axis, the better the solution. Any point below the horizontal line means that the respective algorithm outperformed CPLEX. for the random graphs $ta5$ and $tb5$, as pre-



(a) ta5 (b) tb5

(c) tc5 (d) te4

Figure 5.1: Performance of Greedy Algorithms for 4 large instances

sented in Figure 5.1(a) and 5.1(b), all the greedy algorithms outperformed CPLEX except GA-6,7 and 8. For the geometric graphs $tc5$ and $te4$, as presented in Figure 5.1(c) and Figure 5.1(d), CPLEX reported the optimal solution, all the greedy results report higher establishment cost than CPLEX. Amongst different variations of greedy GA-6,7 and 8 outperformed others.

All the greedy algorithms mentioned in Table 3.1 have been applied to the instances. In general our greedy algorithms are fast, they take few milliseconds to run. Most of greedy algorithms could outperform results generated by CPLEX in all the randomly generated problem instances except two instances - $ta1$ and $tb1$. These instances are relatively smaller than other instances, for which CPLEX took around 3 hours to obtain results (with 94% and 99% integrality gap). Although greedy algorithms could produce results quickly, they could not outperform CPLEX in these two instances. Generally for the rest of randomly gener-

ated problem instances, at least one (or more) of the greedy methods have outperformed CPLEX in terms of establishment cost and running time. GA1 outperformed CPLEX in 6 out of 10 randomly generated problem instances but could not score the lowest establishment cost in comparison with the other greedy algorithms. GA2 outperformed CPLEX in 8 out of 10 randomly generated problem instances and scored 5 lowest establishment cost in comparison with the other greedy algorithms. Performance of GA3 was identical to GA2. GA4 and GA5 performed identically well on the randomly generated problem instances. They outperformed CPLEX in 8 out of 10 random trees and scored the lowest 5 establishment cost values. GA6 performed poorly, it outperformed CPLEX in only two of the instances, and could not score any lowest establishment costs in comparison with other greedy algorithms. GA7 and GA8 performed the worst on the data instances, they did not outperform CPLEX nor other greedy methods. GA9 and GA10 produced identical results and outperform CPLEX in 8 out of 10 random trees and scored 5 lowest establishment costs.

For the problem instances generated from geometric distribution ($tc1 - tc5$, $te1 - te4$) none of the greedy algorithms could get to the optimal solutions obtained by CPLEX. Amongst the greedy algorithms on these instances GA6, GA7 and GA8 performed better than others. GA6 obtained best greedy results on 7 instances out of 9 problem instances generated from geometric distribution. For each of these instances the best results obtained by the greedy algorithms were 5.9% close to the optimal solutions produced by CPLEX on average. The installation costs and running times of different variations of the greedy algorithm on the problem instances have been reported in Table 5.3 and Table 5.4 respectively.

Table 5.3: Greedy Algorithm - Establishment Costs

Problem Instance	Algorithms									
	GA1	GA2	GA3	GA4	GA5	GA6	GA7	GA8	GA9	GA10
ta1	1185	748	748	679	679	1460	1514	1514	679	679
ta2	1511	913	913	915	915	1598	2206	2206	915	915
ta3	1535	987	987	917	917	2022	2613	2613	917	917
ta4	2156	922	922	922	922	1931	2744	2744	922	922
ta5	2045	876	876	1018	1018	2207	3053	3053	1018	1018
tb1	1105	959	959	817	817	1528	1766	1766	817	817
tb2	1287	728	728	728	728	1713	1994	1994	728	728
tb3	1366	919	919	990	990	1698	2091	2091	990	990
tb4	1447	996	996	926	926	1778	2542	2542	926	926
tb5	1870	940	940	870	870	2067	2929	2929	870	870
tc1	4783	4783	4783	4783	4783	4674	4688	4688	4783	4783
tc2	11297	11297	11297	11297	11297	10918	10918	10918	11240	11240
tc3	27297	27297	27297	27297	27297	25906	25901	25901	27180	27180
tc4	60392	60402	60402	60402	60402	56987	56916	56916	59716	59716
tc5	79347	79770	79770	79656	79656	74232	74339	74339	78138	78138
te1	4150	4082	4082	4082	4082	3756	3783	3783	4082	4082
te2	8254	8231	8231	8231	8231	7188	7456	7456	8150	8150
te3	15115	14616	14616	14616	14616	11525	12916	12916	13700	13700
te4	36566	34768	34768	34736	34736	27826	28914	28914	30498	30498

Table 5.4: Greedy Algorithm - Running Time (in 10^{-3} minutes)

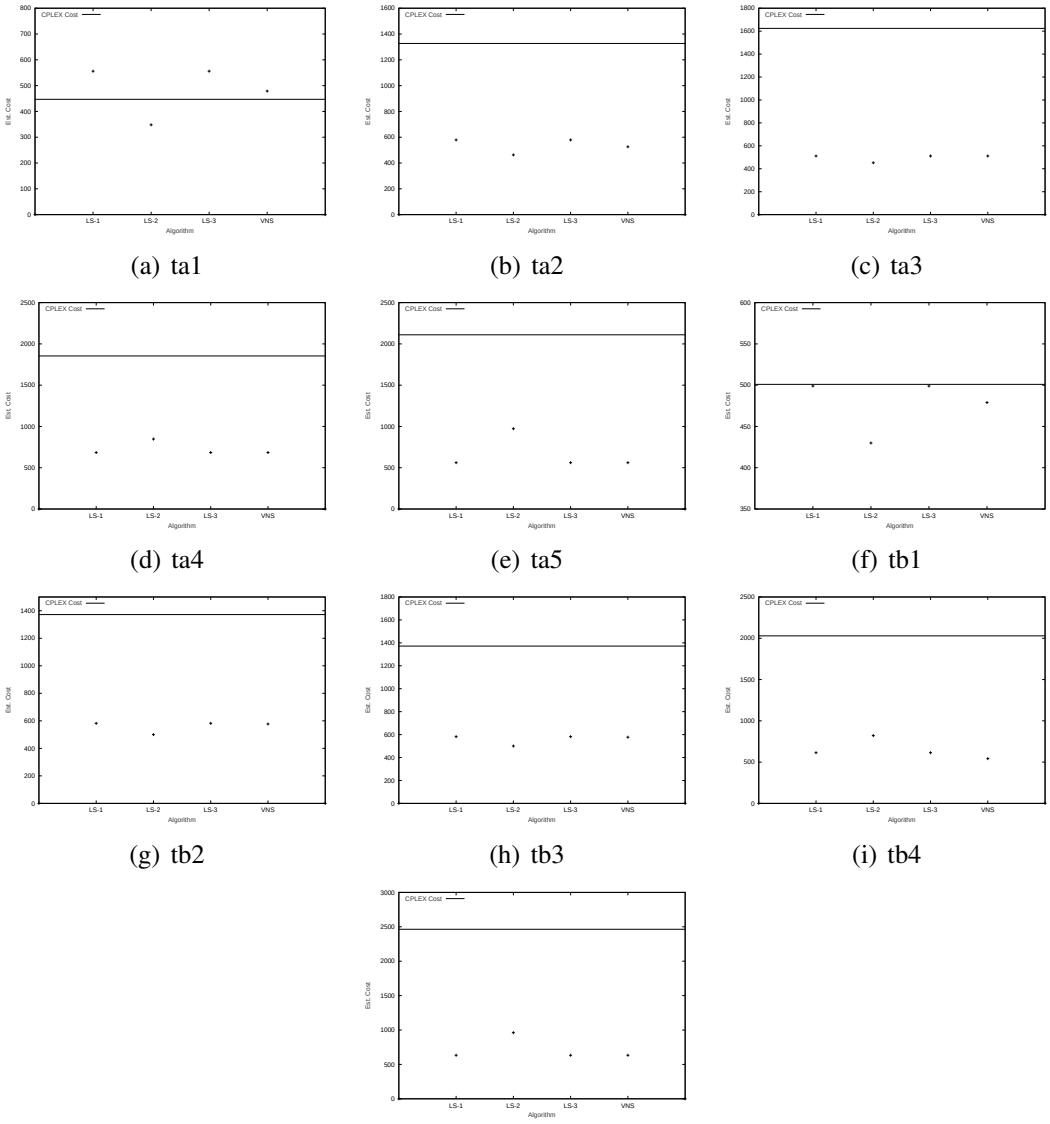
Problem Instance	Algorithms									
	GA1	GA2	GA3	GA4	GA5	GA6	GA7	GA8	GA9	GA10
ta1	0.078	0.078	0.079	0.078	0.077	0.076	0.076	0.077	0.078	0.078
ta2	0.295	0.293	0.293	0.292	0.292	0.293	0.294	0.291	0.296	0.293
ta3	1.156	1.147	1.144	1.149	1.148	1.153	1.145	1.146	1.158	1.158
ta4	4.608	4.600	4.608	4.594	4.587	4.630	4.614	4.609	4.694	4.637
ta5	7.237	7.204	7.241	7.184	7.230	7.187	7.183	7.187	7.322	7.268
tb1	0.170	0.170	0.169	0.169	0.169	0.166	0.166	0.166	0.169	0.168
tb2	0.459	0.453	0.456	0.453	0.453	0.452	0.449	0.453	0.456	0.457
tb3	1.813	1.786	1.779	1.790	1.786	1.781	1.783	1.792	1.797	1.804
tb4	4.614	4.596	4.591	4.620	4.591	4.591	4.585	4.601	4.617	4.635
tb5	7.281	7.226	7.198	7.223	7.222	7.221	7.199	7.199	7.259	7.301
tc1	0.057	0.058	0.058	0.058	0.057	0.057	0.057	0.058	0.057	0.057
tc2	0.221	0.212	0.212	0.213	0.212	0.209	0.212	0.211	0.214	0.215
tc3	0.827	0.818	0.822	0.819	0.818	0.804	0.819	0.811	0.834	0.839
tc4	3.283	3.269	3.274	3.274	3.263	3.209	3.233	3.254	3.308	3.344
tc5	5.083	5.074	5.077	5.089	5.089	4.970	5.023	5.036	5.139	5.141
te1	0.055	0.055	0.055	0.055	0.055	0.054	0.055	0.054	0.055	0.057
te2	0.207	0.205	0.205	0.204	0.205	0.198	0.200	0.201	0.208	0.209
te3	0.803	0.798	0.794	0.792	0.794	0.758	0.768	0.776	0.799	0.806
te4	3.190	3.179	3.169	3.173	3.175	3.017	3.058	3.054	3.190	3.175

5.4 Local Search Algorithms

Algorithms LS-1, LS-2 and LS-3 have a time limit of 180 minutes, after which the algorithm would stop and report the current best solution, otherwise they stop if no improvement on solution is obtained after exhaustively checking their neighbourhoods. The establishment costs obtained by local search algorithms for different problem instances have put in contrast in Figure 5.2 and Figure 5.3. Figure 5.2 presents the establishment costs obtained by local search algorithms in contrast to each other and benchmark obtained by CPLEX. Similar to the previous charts presented on greedy algorithms, the horizontal line represents solution obtained by CPLEX, any point below that line is an improvement over CPLEX, and the lower the point with respect to Y axis the better solution the algorithm has produced. Figure 5.3 presents 4 instances of the geometric graphs, in the rest 5 instanes our algorithms could obtain optimal solution and therefore all the points were falling on the horizontal line representing CPLEX benchmarks.

All the local searches produced significantly better solutions than greedy and CPLEX for instances $ta1 - ta5$ and $tb1 - tb5$, except for two cases: $ta1$ and $tb1$. For problem instance $ta1$, LS-1 and LS-3 have underperformed CPLEX while LS-2 algorithm have outperformed CPLEX producing the smallest establishment cost. For problem instance $tb1$, LS-1 and LS-3 have outperformed CPLEX while LS-2 algorithm have underperformed CPLEX.

For instances $tc1 - tc5$ and $te1 - te4$ all the local searches improved the solution quality when compared to the greedy algorithms. For these instances, CPLEX could find optimal solutions. The local searches also could find the optimal results for some instances, and could get 98.2% close to optimal for the others. In $tc1 - tc4$ and in $te1$, the local search algorithms have produced optimal solutions as produced by CPLEX. In the remaining four of the problem instances generated from geometric distribution, CPLEX outperformed the



(a) ta1 (b) ta2 (c) ta3

(d) ta4 (e) ta5 (f) tb1

(g) tb2 (h) tb3 (i) tb4

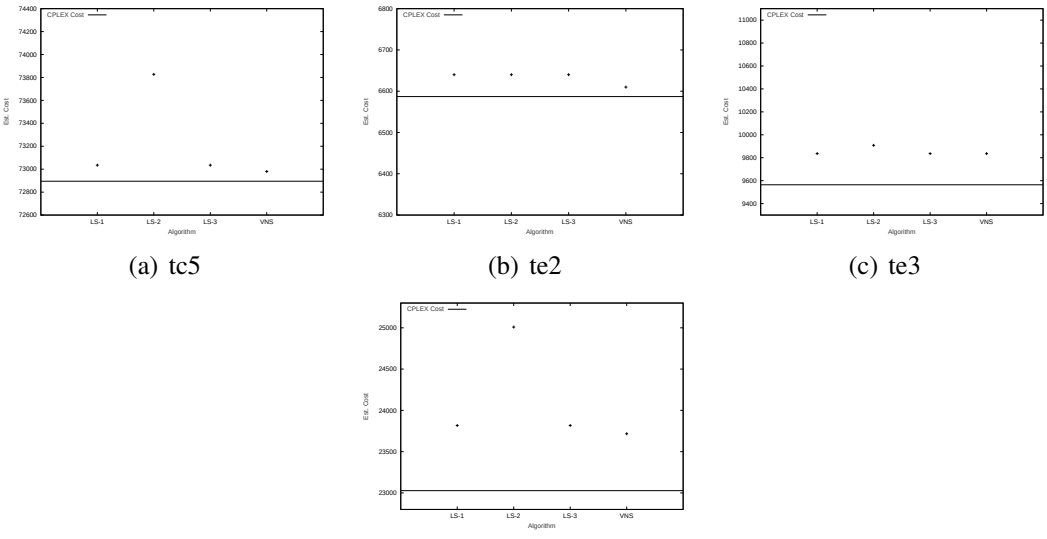
(j) tb5

Figure 5.2: Establishment cost of Local Search Algorithms and VNS for randomly generated problem instances

local search algorithms. However the local search algorithms improved the solutions obtained by the greedy algorithms.

Table 5.5 presents the running times of local search algorithms. As can be seen from LS-2 was always more time consuming than LS-1. However, unless it reached the 180 min limit, LS-2 could produce equal or better solutions than LS-1. This difference is not too evident in case of instances generated from a geometric distribution, but it can be clearly seen for randomly generated instances. The long time needed for LS-2 made it unsuitable for using in a VNS framework. However, it indicated the usefulness of max flow information to find better solution for the problem. LS-3 is an improvement of LS-1 and it obtained better running time for all the instances generated from random geometric distribution On the other hand, it increased the running time for random instances due to the extra precomputation it performs. The data table containing the establishment costs and running times of these algorithms is presented in the appendix of the thesis.

Amongst the three local search algorithms LS-2 seemed to be getting better solutions as soon as it could finish before 180 minutes. This algorithm, however, takes longer running time than the two others and even VNS on both classes of problem instances. This is due to computing max flow for feasibility tests, which makes it computationally more expensive than feasibility tests in the two other local searches and VNS. Although LS-2 have a prohibitive computational time, it gives a strong intuition about the potential of using information from max flow to guide the local search and VNS.



(a) tc5 (b) te2 (c) te3

(d) te4

Figure 5.3: Establishment cost of Local Search Algorithms and VNS for geometric problem instances

Table 5.5: Running time (minutes) of Local Search Algorithms and VNS

Problem Instance	CPLEX	LS-1	LS-2	LS-3	VNS
ta1	180	0.005	1.044	1.011	1.77
ta2	180	0.069	12.267	1.097	1.72
ta3	180	0.288	180.803	2.588	8.199
ta4	180	2.549	180.039	7.219	39.275
ta5	180	13.151	180.021	18.367	111.574
tb1	180	0.018	3.318	1.025	1.166
tb2	180	0.066	47.951	2.108	3.744
tb3	180	1.31	180.008	3.821	19.482
tb4	180	2.246	180.133	7.146	59.631
tb5	180	22.12	180.007	28.057	217.167
tc1	≤ 0.002	0.033	0.0384	1.001	1.764
tc2	≤ 0.002	0.389	0.418	1.249	1.556
tc3	≤ 0.002	24.385	4.962	2.399	6.184
tc4	≤ 0.002	180.011	180.782	12.249	99.303
tc5	≤ 0.002	180.033	180.33	24.254	274.286
te1	≤ 0.002	0.068	0.11	1.781	2.045
te2	≤ 0.002	1.42	1.627	2.701	3.65
te3	0.02	34.551	180	2.881	9.99
te4	0.05	180.01	180.961	19.309	237.014

5.5 Variable Neighborhood Search Algorithms

The installation costs obtained by VNS algorithm on all the instances have been reported in contrast with CPLEX and local search algorithms has been presented in Figure 5.2 and Figure 5.3. The data table is presented in the appendix of this thesis. The running time has been reported in Table 5.5. VNS outperformed the solutions obtained by greedy algorithms in all instances. VNS also outperformed the solutions obtained by CPLEX for randomly generated problem instances in all the cases except for $ta1$, where CPLEX solution was better. In terms of running time on this instance class, VNS recorded better running times than CPLEX (finished running in less than 3 hours), except for the case of tb5, where VNS produced better solutions than CPLEX in a little longer running time. On randomly generated problem instances VNS improved on the solutions obtained by LS-1 and LS-3 in 50% of the cases and produced the same establishment cost on the other 50%. On problem instances generated from geometric distribution, VNS could obtain optimal solutions in 5 cases out of 9. In the other 4 instances (where VNS did not obtain optimal solutions) VNS improved on the results obtained by LS-1 and LS-3 in 3 cases, and produced the same result as LS-1 and LS-3 in the case of $te3$. As a whole on both the problem instance classes, compared to LS-2, VNS outperformed in 47.37% of the cases, produced same installation costs in 26.32% cases, and underperformed in 26.32% cases. It is interesting to observe that LS-2 always produced better results than VNS, except when LS-2 reached 180 min limit. The running time of VNS is longer than that of LS-1 and LS-3 for all the cases, while for a number of instances VNS could finish earlier than LS-2. Compared to its base searching mechanism (LS-3), VNS could produce better results in 60% cases on randomly generated problem instances. For the other instance class it showed improvement in 30% cases. The expense for the qualitative improvement that VNS made was the running time, i.e. VNS algorithm was slowest amongst all proposed algorithms.

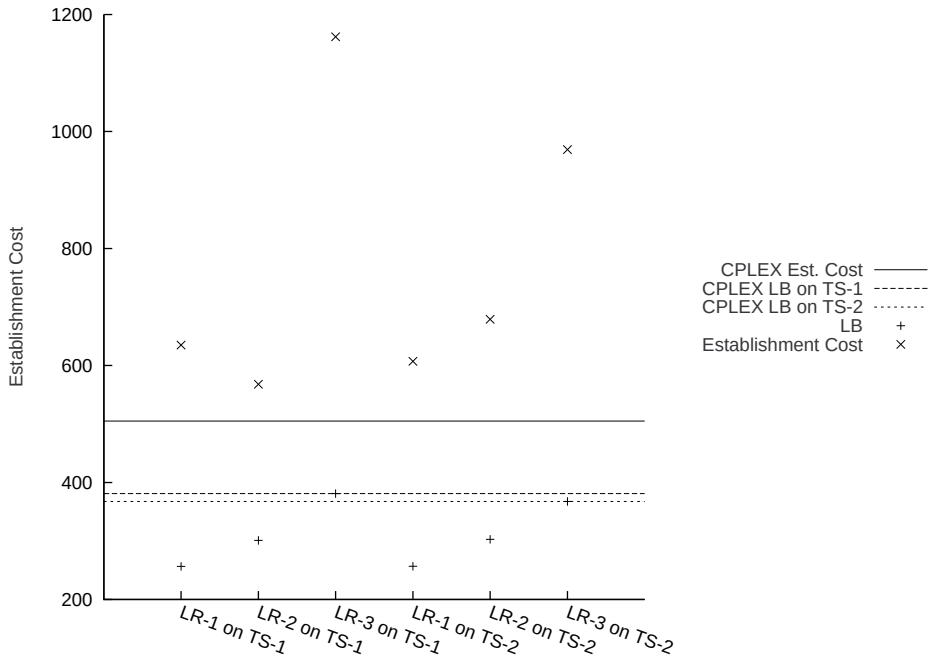


Figure 5.4: Establishment cost and Lower bound (LB) for $ta1$ from different Lagrangian Relaxation approaches

5.6 Lagrangian Relaxation Based Heuristics

We have discussed in Section 4.2 how we can generate a list of trees from the given problem instances. We have generated two set of trees using algorithms described in Section 4.2. The generation process of these tree sets (TreeSet-1, TreeSet-2) have the same initial steps (described in Section 4.2.1). However their generation process differs in the step of generating trees with dual information (described in Section 4.2.2). In the algorithm for generating trees using dual information (Algorithm 7), an input is $(f_b + \pi_b) \forall b \in B$, $(f_r + \pi_r) \forall r \in R$, $\pi_s \forall s \in S$, which is the sum of *tree costs* and Lagrangian multipliers corresponding to all the BSs, RSs and SSs. These Lagrangian multipliers (π_b, π_r, π_s) here can be obtained in two ways. One is using ILOG CPLEX 12.1.0 [92] to solve the model in (4.9-4.11) and use the optimal duals as Lagrangian multipliers. The other is using the iterative process of Lagrangian Relaxation (described in phase 1-2 of Algorithm 10) to obtain the Lagrangian

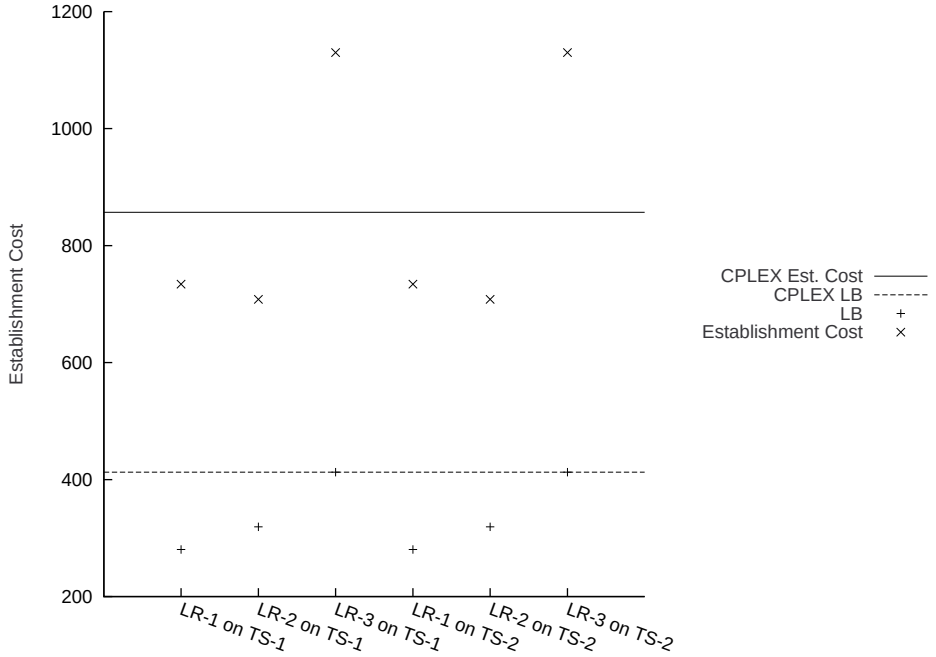


Figure 5.5: Establishment cost and Lower bound (LB) for *ta*2 from different Lagrangian Relaxation approaches

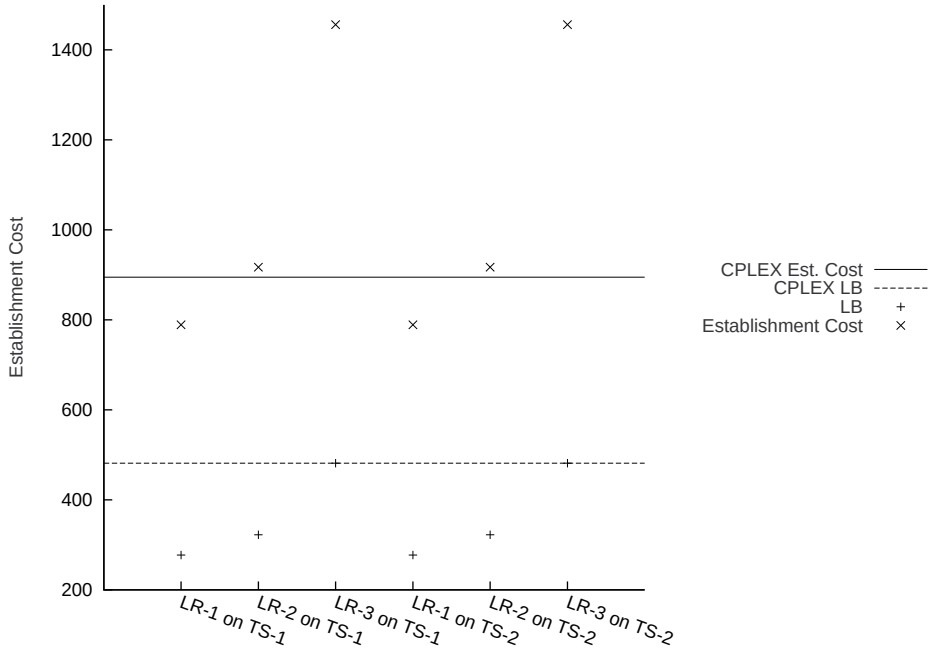


Figure 5.6: Establishment cost and Lower bound (LB) for *ta*3 from different Lagrangian Relaxation approaches

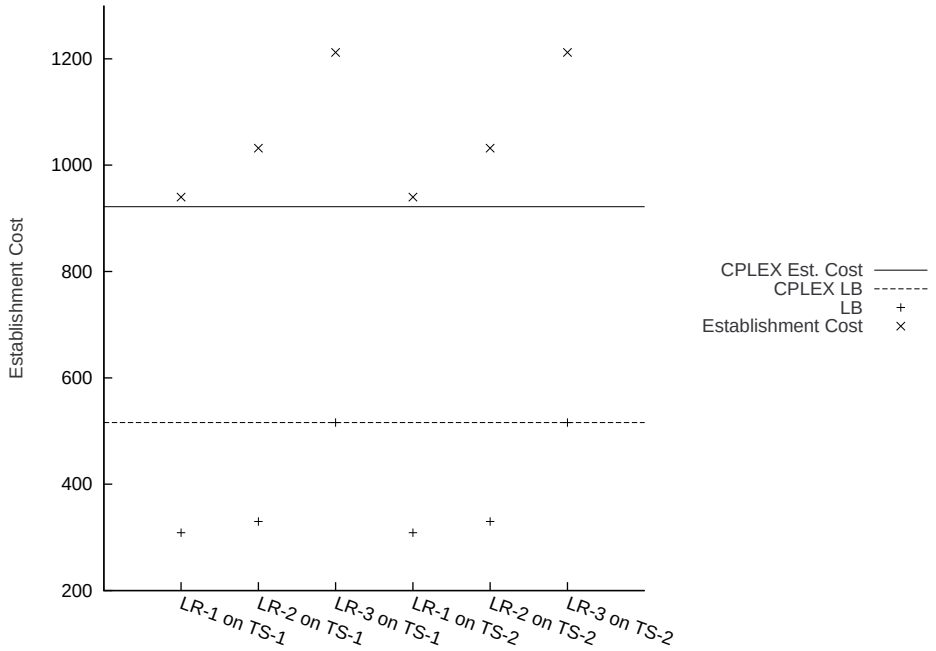


Figure 5.7: Establishment cost and Lower bound (LB) for *ta*4 from different Lagrangian Relaxation approaches

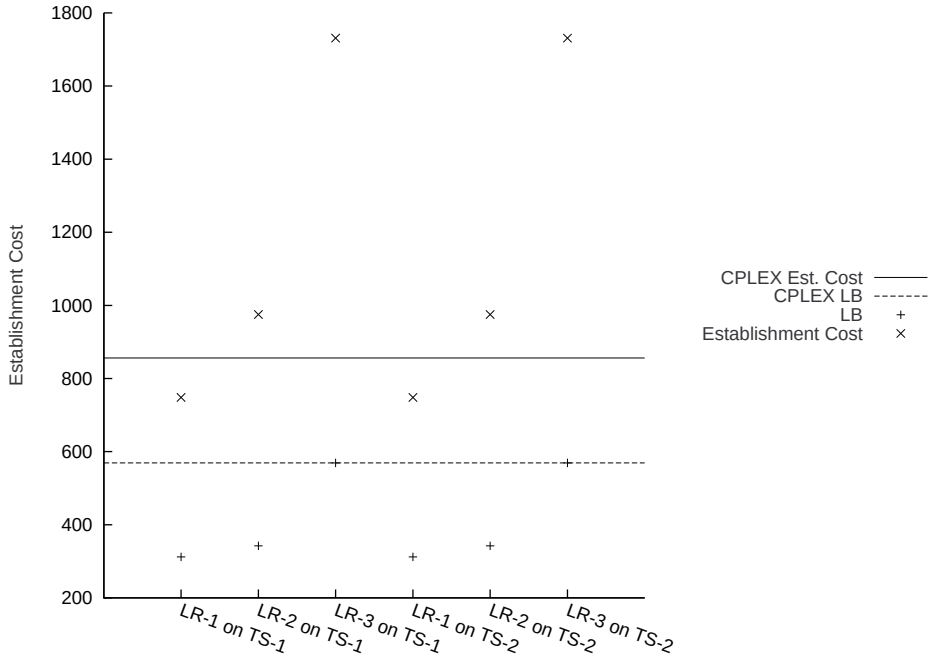


Figure 5.8: Establishment cost and Lower bound (LB) for *ta*5 from different Lagrangian Relaxation approaches

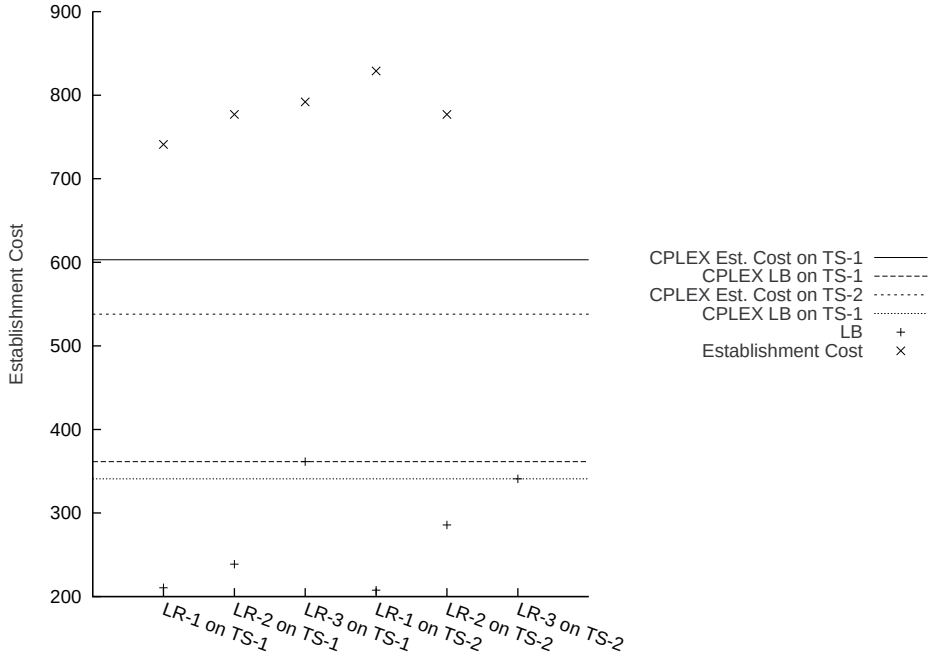


Figure 5.9: Establishment cost and Lower bound (LB) for $tb1$ from different Lagrangian Relaxation approaches

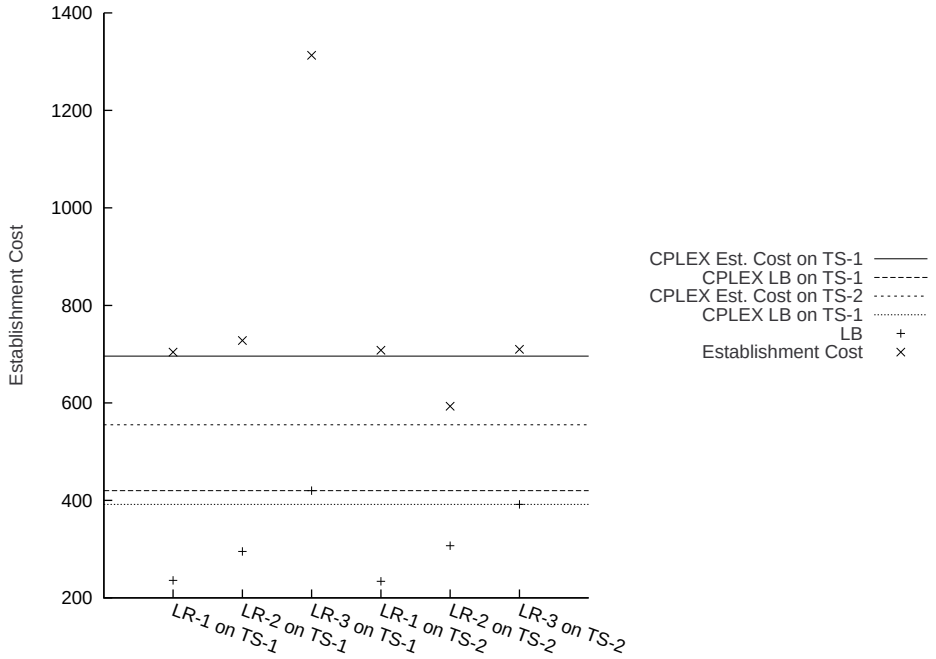


Figure 5.10: Establishment cost and Lower bound (LB) for $tb2$ from different Lagrangian Relaxation approaches

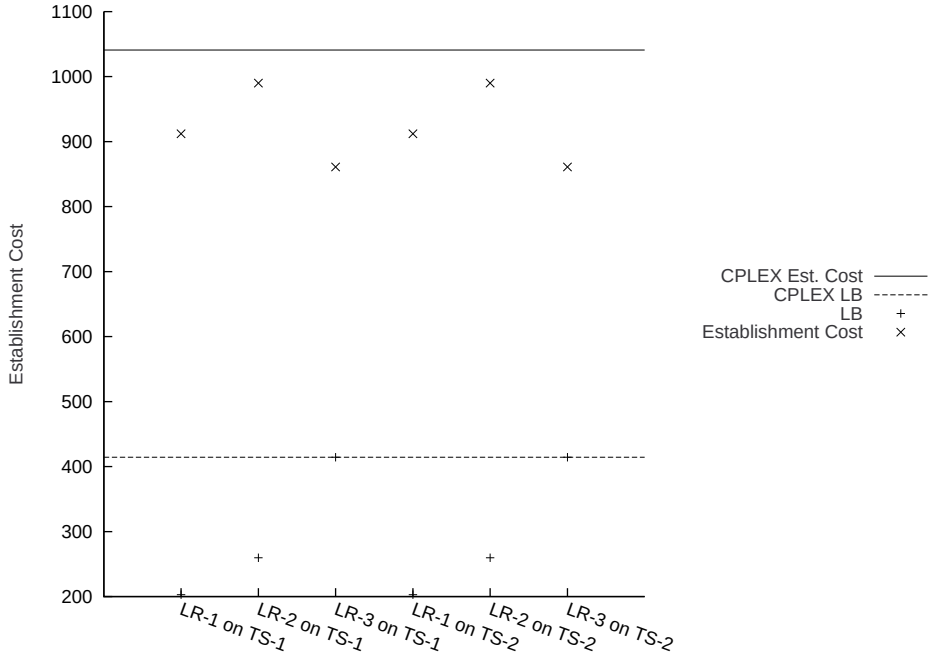


Figure 5.11: Establishment cost and Lower bound (LB) for *tb*3 from different Lagrangian Relaxation approaches

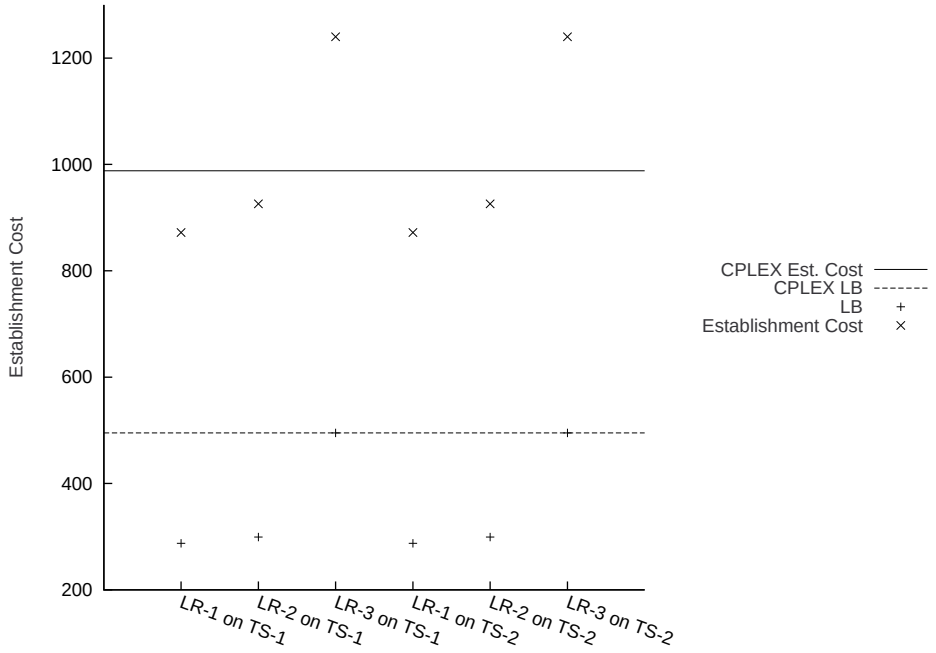


Figure 5.12: Establishment cost and Lower bound (LB) for *tb*4 from different Lagrangian Relaxation approaches

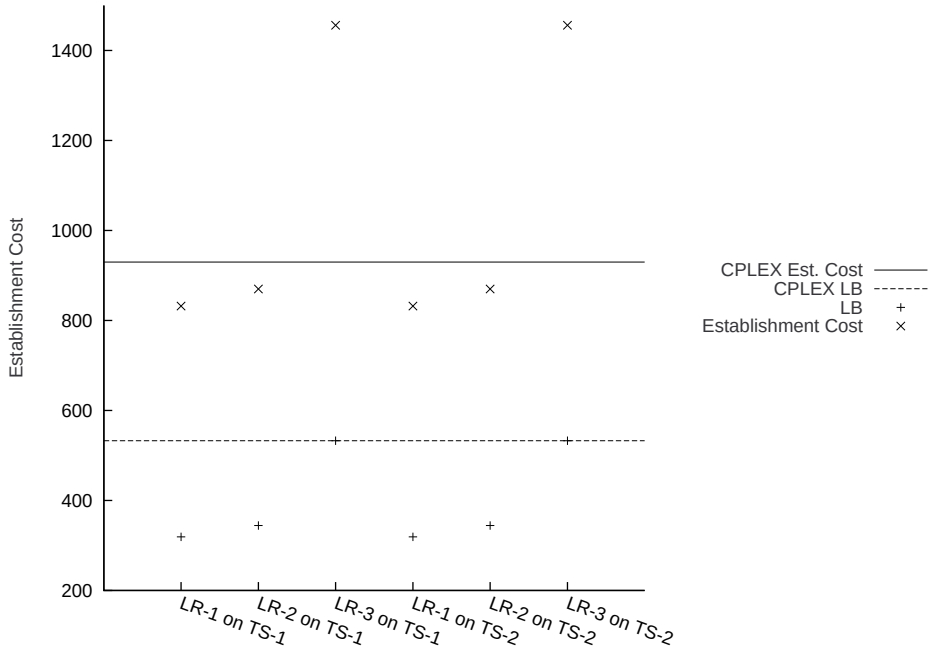


Figure 5.13: Establishment cost and Lower bound (LB) for $tb5$ from different Lagrangian Relaxation approaches

multipliers. To obtain TreeSet-1 we have used optimal duals as π_b, π_r, π_s, and to obtain TreeSet-2 we have used π_b, π_r, π_s from the Lagrangian relaxation process of Algorithm 10 (phase 1-2). Table 5.6 presents the number of trees and total time requited to generate them. On average TreeSet-2 has 21.72% more trees than TreeSet-1, and the generation process of TreeSet-2 took 3.4% more time than that of TreeSet-1.

In Tables 5.7 and 5.8 we report computational results by using ILOG CPLEX 12.1.0 [92] on the tree based approach presented in Chapter 4. All the instances were ran on CPLEX using a time limit of 180 minutes. Most of instances $ta1-ta5$ and $tb1-tb5$ reached 180 minutes and yet had integrality gaps. Instances $tc1-tc5$ and $te1-te4$ finished running and optimal results (with respect to the subset of trees we generated) for them have been reported. Table 5.7 presents the results obtained by CPLEX using TreeSet-1. Table 5.8 presents the results obtained by CPLEX using TreeSet-2. These results will be used to

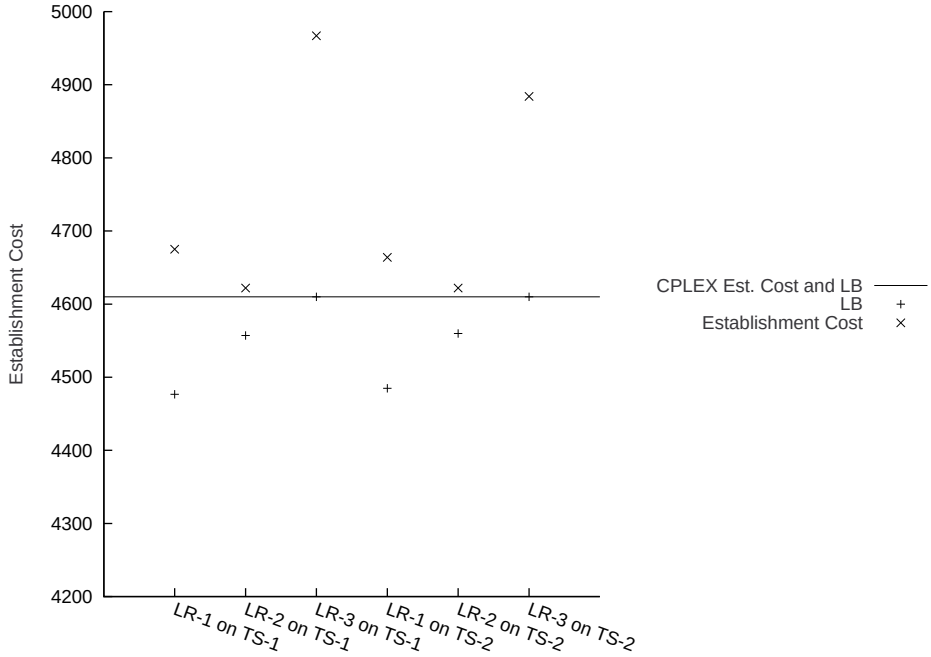


Figure 5.14: Establishment cost and Lower bound (LB) for $tc1$ from different Lagrangian Relaxation approaches

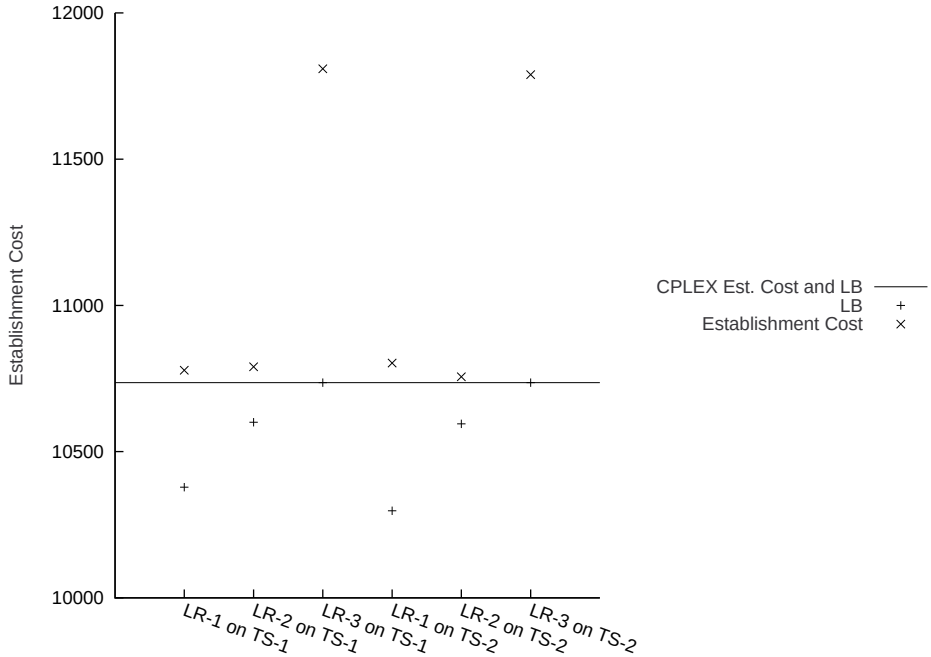


Figure 5.15: Establishment cost and Lower bound (LB) for $tc2$ from different Lagrangian Relaxation approaches

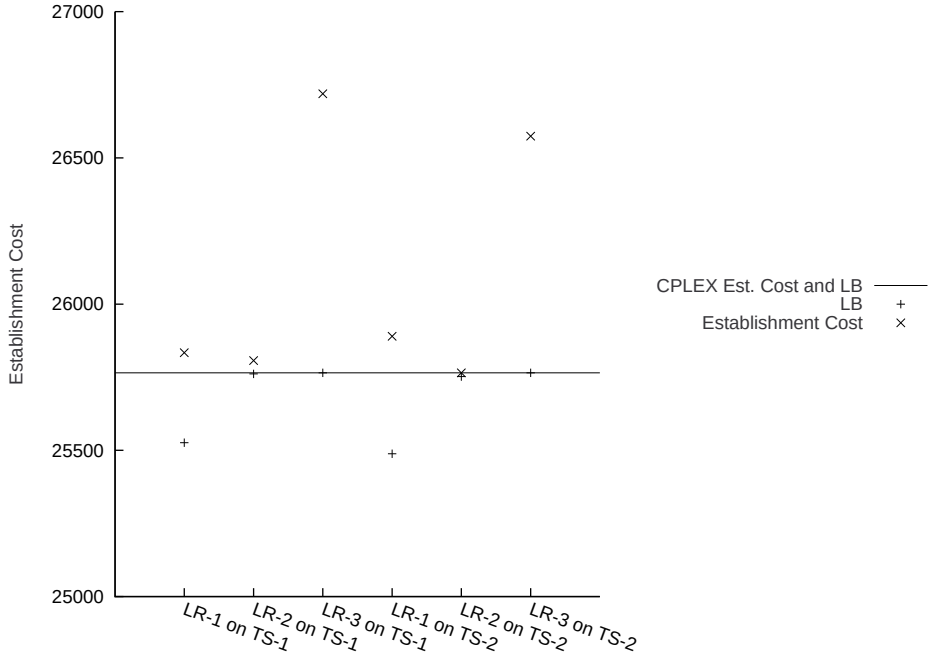


Figure 5.16: Establishment cost and Lower bound (LB) for *tc*3 from different Lagrangian Relaxation approaches

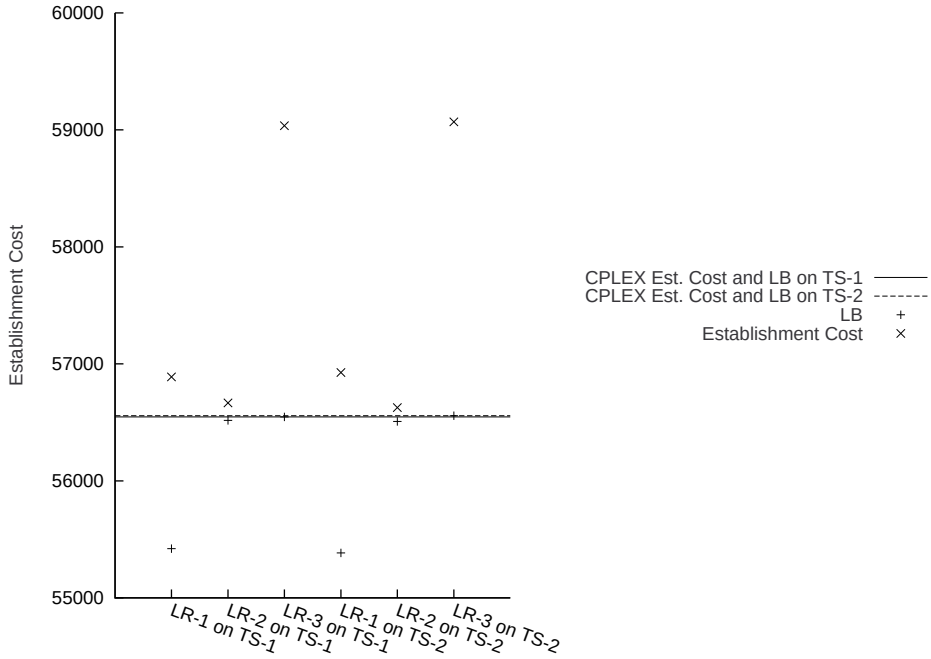


Figure 5.17: Establishment cost and Lower bound (LB) for *tc*4 from different Lagrangian Relaxation approaches

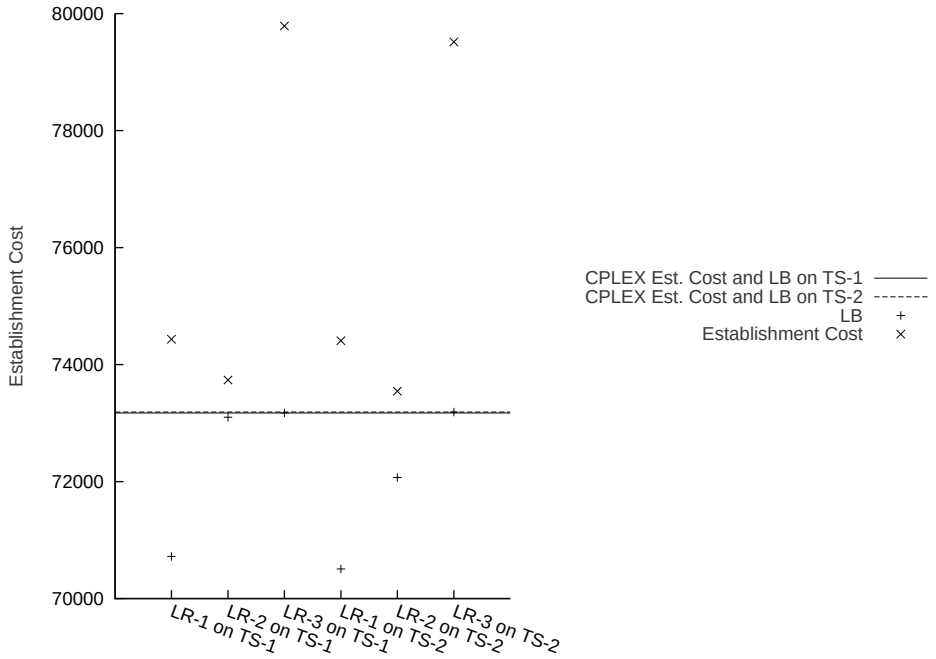


Figure 5.18: Establishment cost and Lower bound (LB) for *tc*5 from different Lagrangian Relaxation approaches

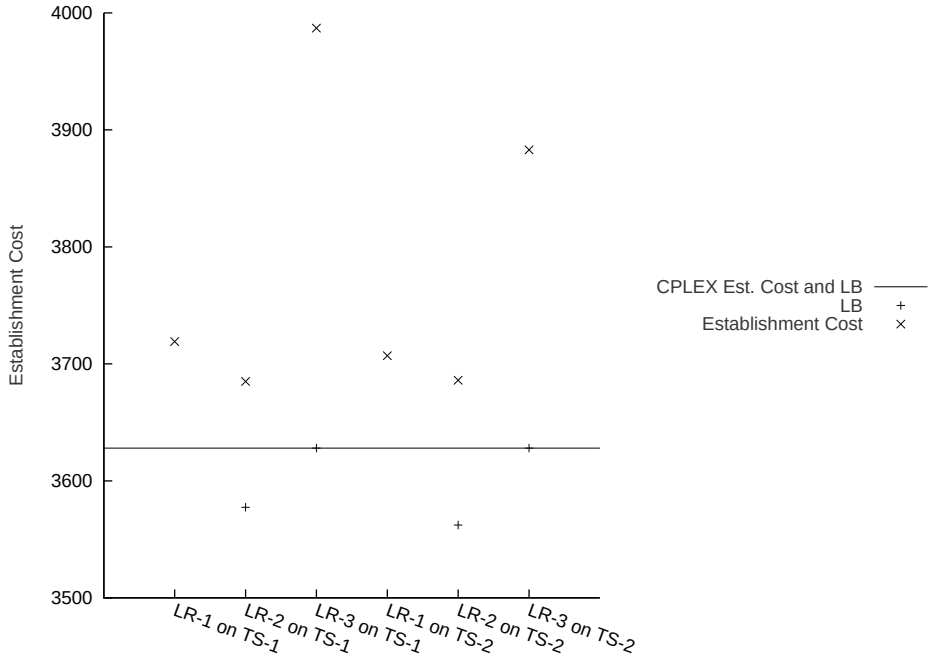


Figure 5.19: Establishment cost and Lower bound (LB) for *te*1 from different Lagrangian Relaxation approaches

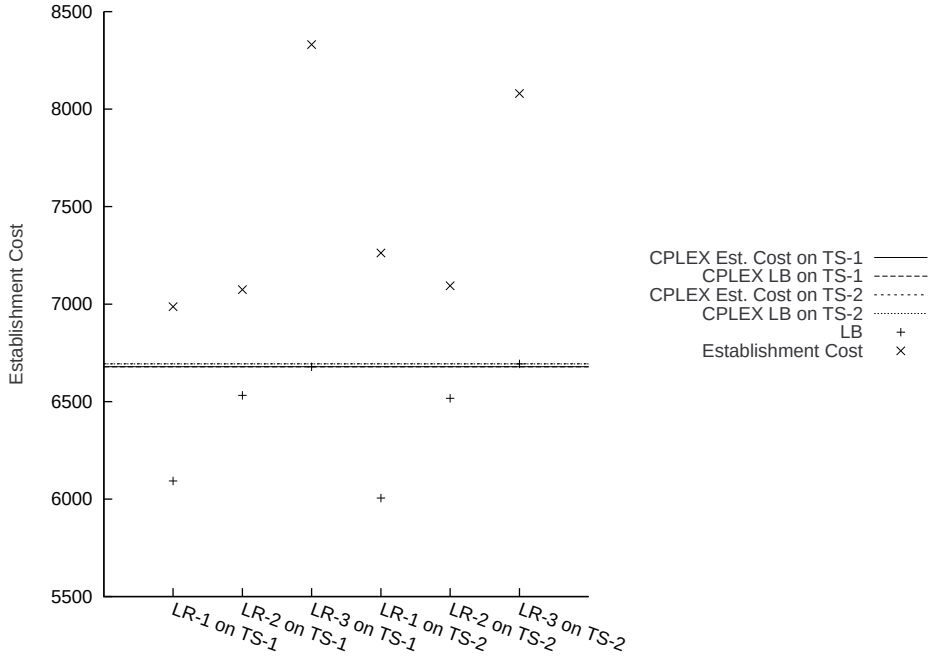


Figure 5.20: Establishment cost and Lower bound (LB) for *te*2 from different Lagrangian Relaxation approaches

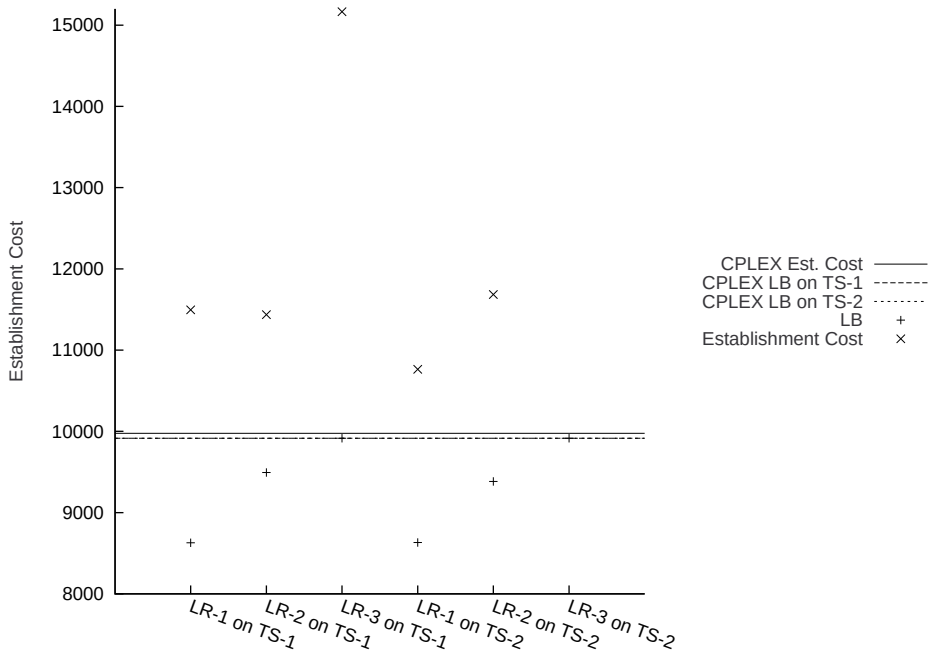


Figure 5.21: Establishment cost and Lower bound (LB) for *te*3 from different Lagrangian Relaxation approaches

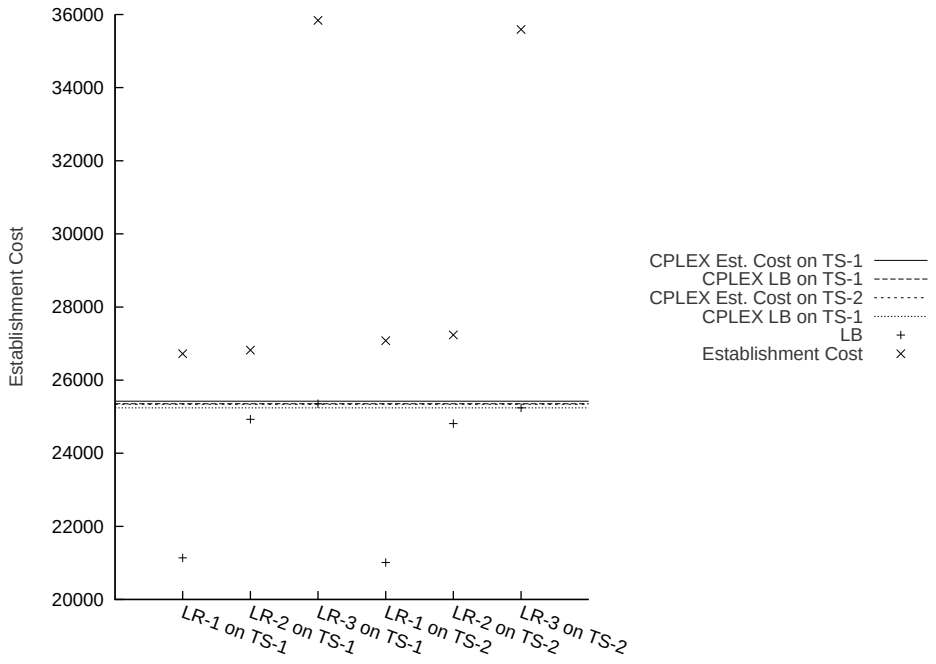


Figure 5.22: Establishment cost and Lower bound (LB) for $te4$ from different Lagrangian Relaxation approaches

compare the results obtained by algorithms presented in Chapter 4.

In Table 5.2 we reported optimal solutions obtained by running instances $tc1-tc5$ and $te1-te4$ in CPLEX with formulation in presented Section 3.1.1. In comparison with those optimal results, the optimal solutions obtained using TreeSet-1 and TreeSet-2 with formulation presented in (4.9-4.11) on the same instances have higher establishment costs. Compared to optimal results reported in Table 5.7 for instances $tc1-tc5$ and $te1-te4$, the optimal solutions obtained using TreeSet-1 as reported in Table 5.7 is 17.36% higher on average. Compared to optimal results reported in Table 5.7 for instances $tc1-tc5$ and $te1-te4$, the optimal solutions obtained using TreeSet-2 as reported in Table 5.8 is 17.27% higher on average.

We have applied our proposed algorithms: LR-1, LR-2, LR-3 (presented in Section 4.3) on tree sets TreeSet-1 and TreeSet-2. These algorithms provided us with a lower bound and

some values of Lagrangian multipliers. We have used these multipliers to obtain a feasible solution combinatorially. For each of the problem instances we plot the lower bounds and establishment costs obtained by our three variations of Lagrangian relaxation on a graph. We also draw horizontal lines with the establishment costs and lower bounds obtained by CPLEX, this helps us measure how close our obtained solutions are in contrast to those obtained by CPLEX on both the tree sets. Figure 5.4 – Figure 5.22 present these graphs for the 19 problem instances. The establishment costs obtained by LR-1, LR-2, LR-3 on both the data sets are higher or equal to the horizontal lines representing CPLEX benchmark, the closer they are to the benchmark lines for establishment costs, the better is the performance of the algorithm. The lower bounds obtained by our algorithms are always lower or equal to the lower bounds obtained by CPLEX. In this case also, the close these points are to the benchmark lower bound lines, the better is the performance of the algorithm. In some cases, especially for geometric graphs, the benchmark lines overlapped. We have provided the establishment cost and the lower bound data obtained by LR-1, LR-2, LR-3 in the appendix of this thesis.

The algorithm LR-3 uses solution of model in equation (4.9-4.11) as Lagrangian multipliers to obtain feasible solution. The lower bound in case of LR-3 is same as the lower bound obtained by CPLEX as presented in Table 5.7 and Table 5.8 for TreeSet-1 and TreeSet-2 respectively. Algorithm LR-2 obtains better lower bounds than Algorithm LR-1 for all the input instances for both TreeSet-1 and TreeSet-2. On instances generated from geometric distribution, LR-2 produced better lower bounds for TreeSet-1 than in case of TreeSet-2. On randomly generated instances, LR-2 performed similar on TreeSet-1 and TreeSet-2, it produced the same lower bounds in 7 out of 10 randomly generated instances, on the rest 3 LR-2 on TreeSet-2 produced 6.92% better lower bounds on average.

The establishment costs obtained by LR-1, LR-2 and LR-3 are presented in Figure 5.4 – Figure 5.22 on TreeSet-1 and TreeSet-2. On instances generated from geometric distri-

bution, LR-2 produced the best lower bounds for TreeSet-2 on 7 out of 9 such instances. For randomly generated instances LR-1 obtained best lower bound on TreeSet-1 in 7 cases out of 10. LR-3 performed poorly on both the tree sets on both set of input instances.

Table 5.9 presents the run times for LR-1 LR-2 and LR-3 on TreeSet-1 and TreeSet-2. LR-2 recorded the best timings for both TreeSet-1 and TreeSet-2. On randomly generated instances LR-2 on TreeSet-2 recorded the best timing in 9 out of 10 instances, with an average 3.21% run time improvement compared to run time of LR-2 on TreeSet-1. On instances generated from geometric distribution LR-2 on TreeSet-1 recorded the best timing on all instances compared to run time of LR-2 on TreeSet-2. For these instances LR-2 on TreeSet-2 also obtained same timings as LR-2 on TreeSet-1 on 7 instances out of 9. Compared to LR-3 on TreeSet-1, LR-3 TreeSet-2 scored better timings on 8 out of 9 instances generated from geometric distribution. LR-3 obtained an average 58.41% run time improvement on TreeSet-2 compared to its running time on TreeSet-1. LR-1 on both tree sets scored the longest running times.

To observe how different the Lagrangian multipliers π_{LR} obtained by our Lagrangian algorithms with respect to Lagrangian multipliers obtained from optimal dual variables π_{CPLEX}, we calculate the ratio between their difference and π_{CPLEX} as follows: $||\pi_{CPLEX} - \pi_{LR}||/||\pi_{CPLEX}||$. We call it π-difference ratio. Table 5.10 presents this ratio for LS-1 and LS-2 on both TreeSet-1 and TreeSet-2. Since LR-3 uses optimal dual variables obtained by CPLEX, this ratio for LR-3 is 0, and we do not report them in this table. For randomly generated instances, the Lagrangian multipliers closest to optimal dual variables obtained by CPLEX is obtained by LR-2 on TreeSet-2, in 9 of these instances out of 10 LR-2 on TreeSet-2 scored the lease gap. For instances generated over a geometric distribution, LR-2 on TreeSet-2 scored the best ratios in 5 instances out of 9.

Table 5.6: Two Set of trees and their generation time (minutes)

Problem	TreeSet-1		TreeSet-2	
Instance	#Trees	Generation time	#Trees	Generation time
ta1	735	0.442	858	0.449
ta2	971	3.815	1038	3.996
ta3	1089	21.339	1521	22.488
ta4	1430	81.964	1709	90.957
ta5	1633	130.778	2022	141.129
tb1	871	1.695	1015	1.789
tb2	870	5.164	1153	5.318
tb3	1307	45.097	1617	47.361
tb4	1432	79.743	1690	88.607
tb5	1664	112.131	2184	117.508
tc1	298	0.033	329	0.033
tc2	568	0.049	631	0.049
tc3	800	0.114	968	0.115
tc4	1710	0.390	2037	0.393
tc5	2657	0.601	2806	0.606
te1	428	0.027	508	0.027
te2	1613	0.087	2424	0.087
te3	5264	0.594	5251	0.603
te4	6879	1.098	10191	1.101

Table 5.7: Results from CPLEX for tree based approach on TreeSet-1

Instance Type	Problem Instance	Establishment Costs	Integrality Gap	Lower Bound	Running Time (Min.)
Random Graphs	ta1	505	0.95%	380.996	0.216
	ta2	857	32.28%	412.704	180.035
	ta3	895	30.09%	481.492	180.048
	ta4	922	42.15%	515.92	180.084
	ta5	856	17.24%	569.263	180.06
	tb1	603	0.40%	361.531	7.294
	tb2	696	0.40%	419.921	32.532
	tb3	1041	56.11%	414.292	180.027
	tb4	988	47.26%	495.236	180.043
	tb5	930	39.00%	532.748	180.088
Geometric Graphs	tc1	4610	0.00%	4610	$\leq$0.001
	tc2	10736	0.00%	10736	$\leq$0.001
	tc3	25765	0.00%	25765	$\leq$0.001
	tc4	56547	0.00%	56547	$\leq$0.001
	tc5	73174	0.00%	73174	$\leq$0.001
	te1	3628	0.01%	3628	$\leq$0.001
	te2	6679	0.01%	6678.5	0.002
	te3	9976	0.04%	9915.39	0.008
	te4	25424	0.00%	25354.2	0.008

Table 5.8: Results from CPLEX for tree based approach on TreeSet-2

Instance Type	Problem Instance	Establishment Costs	Integrality Gap	Lower Bound	Running Time (Min.)
Random Graphs	ta1	505	1.61%	367.516	0.242
	ta2	857	34.66%	412.704	180.12
	ta3	895	30.97%	481.492	180.063
	ta4	922	42.17%	515.92	180.044
	ta5	856	32.15%	569.263	180.056
	tb1	538	0.52%	340.9	2.896
	tb2	555	5.63%	391.768	0.582
	tb3	1041	55.44%	414.292	180.027
	tb4	988	46.42%	495.236	180.056
	tb5	930	39.00%	532.748	180.002
Geometric Graphs	tc1	4610	0.00%	4610	$\leq$0.001
	tc2	10736	0.00%	10736	$\leq$0.001
	tc3	25765	0.00%	25765	$\leq$0.001
	tc4	56557	0.00%	56557	$\leq$0.001
	tc5	73190	0.00%	73190	$\leq$0.001
	te1	3628	0.00%	3628	$\leq$0.001
	te2	6694	0.00%	6693.5	$\leq$0.001
	te3	9976	0.01%	9915.39	0.008
	te4	25340	0.03%	25239.3	0.011

5.7 Summary

The local search based algorithms presented in chapter 4 could outperform the CPLEX benchmarks for class $ta1-ta5$ and $tb1-tb5$ of problem instances in smaller computational times. For instance class $tc1-tc5$ and $te1-te4$ our algorithms produced optimal result in

Table 5.9: Running time (minutes) for different Lagrangian relaxation algorithms

Problem	TreeSet-1			TreeSet-2		
Instance	LR-1	LR-2	LR-3	LR-1	LR-2	LR-3
ta1	0.097	0.002	0.013	0.098	0.002	0.004
ta2	0.390	0.005	0.068	0.389	0.004	0.017
ta3	1.398	0.009	0.189	1.380	0.009	0.038
ta4	6.078	0.044	0.628	5.969	0.041	0.166
ta5	10.535	0.061	1.206	11.670	0.057	0.415
tb1	0.259	0.006	0.031	0.356	0.006	0.008
tb2	0.327	0.006	0.064	0.431	0.007	0.011
tb3	4.937	0.023	0.265	6.754	0.021	0.072
tb4	6.262	0.049	0.581	9.429	0.044	0.164
tb5	9.402	0.073	0.832	15.291	0.067	0.212
tc1	0.003	0.181×10^{-3}	0.001	0.003	0.020×10^{-3}	0.195×10^{-3}
tc2	0.010	0.001	0.002	0.018	0.001	0.001
tc3	0.039	0.001	0.003	0.045	0.001	0.001
tc4	0.171	0.005	0.015	0.188	0.006	0.005
tc5	0.287	0.008	0.027	0.297	0.008	0.014
te1	0.003	0.272×10^{-3}	0.001	0.003	0.361×10^{-3}	0.235×10^{-3}
te2	0.019	0.001	0.002	0.029	0.001	0.001
te3	0.273	0.005	0.009	0.333	0.009	0.014
te4	0.623	0.011	0.031	0.675	0.011	0.017

Table 5.10: π-difference ratio (in %) for different Lagrangian relaxation algorithms

Problem	TreeSet-1		TreeSet-2	
Instance	LR-1	LR-2	LR-1	LR-2
ta1	93.594	90.247	95.188	88.473
ta2	95.006	91.456	95.010	91.451
ta3	98.988	96.941	99.387	96.941
ta4	100.531	99.242	100.531	99.143
ta5	99.471	98.465	99.471	98.465
tb1	101.033	92.400	99.279	83.891
tb2	98.559	95.188	99.832	93.245
tb3	107.033	100.386	106.858	100.373
tb4	98.887	98.334	99.166	98.306
tb5	98.587	97.833	98.587	97.776
tc1	76.481	75.843	77.481	76.697
tc2	80.148	79.323	80.704	79.591
tc3	81.075	80.811	81.312	80.887
tc4	79.638	79.226	79.948	79.003
tc5	80.918	79.535	80.221	78.970
te1	81.079	79.573	80.587	77.522
te2	78.712	75.415	81.580	78.197
te3	86.911	83.458	85.101	81.954
te4	89.671	85.655	87.876	82.300

50% of the cases and results with worst gap of 3% to optimal solutions produced by CPLEX on the other cases.

A big limitation of the local search and VNS algorithm is the exponential nature of run time increase with respect to growth in problem instance size. This still remains an issue to be addressed in order to use these algorithms to handle larger problem instances. Future work to improve these algorithms is directed towards: exploiting the information that can be available from max flow to guide the local search / VNS procedure without using max flow for feasibility checking, and using more suitable data structures to reduce computational time.

For the Lagrangian algorithms LR-3 produced the best lower bounds, however it uses CPLEX to obtain the Lagrangian multipliers. LR-2 performed better amongst the Lagrangian algorithms that does not use CPLEX. LR-2 performed better on TreeSet-1 than TreeSet-2. The lower bounds obtained by LS-2 on TreeSet-2, although not as good as those of LS-2 on TreeSet-1, produces close scores to the corresponding TreeSet-1. Lower bounds obtained by LS-2 on TreeSet-2 are better than lower bounds obtained by LR-1.

In terms of running time and π-difference ratios, LR-2 on TreeSet-2 outperforms all others. If we consider a situation when we do not have access to CPLEX, LR-2 provides good running times, lower bounds and π-difference ratios.

In terms of establishment costs, LR-1 outperformed LR-2 and LR-3. It is interesting to observe that LR-3 (although using the best Lagrangian multipliers obtained from CPLEX dual variables) is not producing the best establishment costs. This gives us intuition towards scope of improvement in the phase 3 of Algorithm 8 and Algorithm 10.

Use of the iterative process of Lagrangian algorithm to obtain TreeSet-2 produced more trees than what we got from using optimal dual from CPLEX from tree generation (as we did to produce TreeSet-1). We used a greedy procedure to generate trees in this step using $(f_b + \pi_b) \forall b \in B$, $(f_r + \pi_r) \forall r \in R$, $\pi_s \forall s \in S$. The tree set with higher number of trees

(TreeSet-2) led to quickly produce good π-difference ratios, lower bounds and establishment costs. Another potential of enhancing these scores lie in producing better tree set by smarter use of the Lagrangian multipliers.

Chapter 6

Conclusion and Future Work

In this thesis we have presented a new network planning problem in IEEE 802.16j wireless mesh network. We have also formulated several mathematical models to represent the problem and analyzed them. The understanding of algorithms and approaches applied to solve the four NP hard problems as presented in Chapter 2 (Set Cover, Capacitated Set Cover, Bin Packing and Unsplittable Flow) helped to develop insight into dealing with planning and design problems in WiMAX, some of which we have reviewed in the introduction. From a general perspective, design or planning of almost all wireless network (WiMAX, WiFi, sensor network, etc) comes with the inherent limitation on capacity of equipment, location, bandwidth, time or channel allocation, quality of service, scheduling, etc. Our study on the four problems puts light on how the researchers of theoretical computer science have handled similar problems and how their solutions and contributions can be helpful to solve design or engineering problems of wireless network.

We have established that our research problem is NP hard by showing a polynomial time reduction of Bin Packing to a restricted version of WiMAX planning problem. Since there is no known polynomial time algorithm to solve NP hard problems, the optimal solution to our problem also cannot be produced in polynomial time. If the problem instances have few BS, RS and SS, we can exhaustively enumerate the possible solutions and pick the best result. But when the input instances have large number of BS, RS and SS, exhaustive search for the best solution would take exponentially large time with respect to the input size. In this respect application of heuristics/meta heuristics based solution space search to approximate a near-optimal result in reasonable computational time and resource seemed to be intuitively the most promising approach for solving the problem. As a starting point, we implemented local search and variable neighborhood search.

We have experimented on the problem using heuristics based on different variations of local search and a variable neighborhood search. We have tested the algorithms we developed with different randomly generated network instances of size 500-5000 nodes. We compared the results obtained by our algorithms with numerical optimization problem solver called CPLEX [92]. Our algorithms could obtain optimal solution for a number of input instances, find better solutions than obtained by CPLEX for some others, and obtained a worst case integrality gap of 3% for the rest of the instances. These experiments and their results have been reported in [101].

We have also modeled the WiMAX planning problem in a way similar to set covering. This model has been used to apply Lagrangian relaxation. We have applied variations of Lagrangian relaxation algorithms and analyzed the performance. Amongst the three implementation of Lagrangian algorithms we implemented one has a fast running time and thereby has potential to be used with other approaches on the problem. Also these algorithms can be used in a framework where we fix part of the solution obtained by the Lagrangian algorithms, and re-optimize the rest of the problem.

We have discussed in Chapter 5 that there is scope of improvement in the process of obtaining a feasible solution to the problem (phase 3 of Algorithm 8 and Algorithm 10) given values for Lagrangian multipliers. The next idea we are interested to try is tree fixing and re-optimization. This means, given an instance of the problem, we will generate trees using greedy, local search and Lagrangian relaxation, and use the last stage of Lagrangian relaxation to obtain the Lagrangian multipliers, as we do with the algorithms presented in Section 4.2 and 4.3. With these Lagrangian multipliers we will calculate the reduced cost for all the available trees, and pick the tree with the least reduced cost. We will fix this tree as a part of the solution, i.e. will take the RSs and BSs in the tree in the solution. We will also allocate the SSs (which are in the tree) in the solution to these RSs and BSs in exactly same fashion as it was in the tree. Than we remove the SSs, RSs and BSs of this tree from

the problem instance, and re-optimize this reduced size instance using tree generation from greedy, local search and Lagrangian relaxation, use the last stage of Lagrangian relaxation to obtain the Lagrangian multipliers to further select trees with least reduced cost to fix it in the solution. This process stops when all the demand of SS are served by RSs/BSs.

Since we proposed this WiMAX planning problem and could not find from the available literature any real data to exactly fit the problem, we have used synthetic data to experiment and validate efficiency of our proposed algorithms. It could be interesting to see how our algorithms perform on real world network data, that would solidify our observations from experimental results, and might strengthen our claims in the thesis.

Future research on the WiMAX planning problem can be driven towards a number of directions. The first is development of approximation algorithms with better upper and lower bound guarantee. We can use the Lagrangian relaxation process to guide the local searches to obtain solutions with a lower bound guarantee. The second could be studying the significance of the solution to the corresponding max flow problem on the optimal/near optimal of WiMAX planning problem. We have seen a similar approach taken on UFP. Thirdly, we have developed integer programming (IP) formulations of the WiMAX planning problem in [101]. It would be nice to study the relaxations of the IPs, and use primal-dual based procedure on the problem.

We have applied VNS on the problem and reported results in [101]. It would be interesting to see how variable neighborhood descent (VND) and variable neighborhood decomposition search performs on the planning problem. We applied VND to the WiMAX planning problem in earlier phases of our research, but it was slow in terms of running time, and the solutions obtained from this algorithm was not as good as VNS considering their comparative running times. This is why we did not use VND based approach in our later phases of research. But this framework can be useful to quickly generate trees using Lagrangian relaxation method, for example. The idea is to look into the limited scope of some

initial trees, calculate their Lagrangian multipliers, and thereby greedily generate trees with negative reduced cost in that confined scope. If or when no more trees can be generated in that initial scope, the scope of the trees is enlarged, and thereby generating more tree using Lagrangian multipliers and reduced costs calculations. This enlarging of scope continues until some stopping condition is met.

Appendix A

Additional Experimental Data

Table A.1: Local Search Algorithm

Problem Instance	LS-1		LS-2		LS-3	
	Estab. cost	Running Time (Min)	Estab. cost	Running Time (Min)	Estab. cost	Running Time (Min)
ta1	556	0.005	348	1.044	556	1.011
ta2	579	0.069	463	12.267	579	1.097
ta3	511	0.288	452	180.803	511	2.588
ta4	684	2.549	847	180.039	684	7.219
ta5	561	13.151	973	180.021	561	18.367
tb1	499	0.018	430	3.318	499	1.025
tb2	582	0.066	500	47.951	582	2.108
tb3	653	1.31	787	180.008	653	3.821
tb4	614	2.246	821	180.133	614	7.146
tb5	633	22.12	961	180.007	633	28.057
tc1	4610	0.033	4610	0.0384	4610	1.001
tc2	10724	0.389	10724	0.418	10724	1.249
tc3	25678	24.385	25678	4.962	25678	2.399
tc4	56464	180.011	56464	180.782	56464	12.249
tc5	73035	180.033	73827	180.33	73035	24.254
te1	3618	0.068	3618	0.11	3618	1.781
te2	6640	1.42	6640	1.627	6640	2.701
te3	9836	34.551	9908	180	9836	2.881
te4	23817	180.01	25009	180.961	23817	19.309

Table A.2: VNS Algorithm

Problem Instance	Establishment Cost	Running time (Min)
ta1	479	1.770
ta2	526	1.720
ta3	511	8.199
ta4	684	39.275
ta5	561	111.574
tb1	479	1.166
tb2	577	3.744
tb3	487	19.482
tb4	542	59.631
tb5	633	217.167
tc1	4610	1.764
tc2	10724	1.556
tc3	25678	6.184
tc4	56464	99.303
tc5	72980	274.286
te1	3618	2.045
te2	6610	3.650
te3	9836	9.990
te4	23717	237.014

Table A.3: Lower bound for different Lagrangian relaxation algorithms

Problem	TreeSet-1		TreeSet-2	
Instance	LR-1	LR-2	LR-1	LR-2
ta1	256.573	301.019	256.842	302.81
ta2	280.551	319.307	280.551	319.307
ta3	277.353	322.251	277.353	322.251
ta4	308.9	329.863	308.9	329.863
ta5	311.847	342.4	311.847	342.4
tb1	210.604	238.83	207.724	285.688
tb2	235.935	295.287	233.921	306.905
tb3	202.857	259.768	202.857	259.768
tb4	287.425	299.297	287.425	299.297
tb5	319.227	344.226	319.227	344.226
tc1	4476.62	4557.01	4484.76	4559.76
tc2	10378.2	10600.2	10297.5	10595.1
tc3	25526	25761.5	25488.2	25752.4
tc4	55420.1	56516.8	55384.1	56507.7
tc5	70720.5	73099.8	70506.9	72070.6
te1	3385.43	3577.41	3380.9	3562.25
te2	6092.87	6531.59	6005.15	6516.79
te3	8627.81	9493.2	8631.89	9384.06
te4	21137.9	24928.2	21009.6	24810

Table A.4: Establishment cost for different Lagrangian relaxation algorithms

Problem	TreeSet-1			TreeSet-2		
Instance	LR-1	LR-2	LR-3	LR-1	LR-2	LR-3
ta1	635	568	1162	607	679	969
ta2	734	708	1130	734	708	1130
ta3	789	917	1456	789	917	1456
ta4	940	1032	1212	940	1032	1212
ta5	748	975	1731	748	975	1731
tb1	741	777	792	829	777	922
tb2	704	728	1313	708	593	710
tb3	912	990	861	912	990	861
tb4	872	926	1240	872	926	1240
tb5	832	870	1456	832	870	1456
tc1	4675	4622	4967	4664	4622	4884
tc2	10778	10790	11809	10803	10756	11789
tc3	25834	25807	26719	25890	25765	26574
tc4	56889	56666	59036	56926	56626	59069
tc5	74434	73735	79789	74408	73544	79517
te1	3719	3685	3987	3707	3686	3883
te2	6987	7074	8331	7262	7094	8080
te3	11496	11437	15165	10764	11685	15413
te4	26723	26820	35841	27078	27237	35590

Bibliography

[1] E.H.L. Aarts and J.K. Lenstra. *Local search in combinatorial optimization*. Princeton Univ Pr, 2003.

[2] D. Abusch-Magder. Novel algorithms for reducing cell sites during a technology upgrade and network overlay. In *Wireless Communications and Networking Conference, 2005 IEEE*, volume 3, pages 1726–1732. IEEE, 2005.

[3] R. Aharoni. Ryser's conjecture for tripartite 3-graphs. *Combinatorica*, 21(1):1–4, 2001.

[4] I.F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. *Computer networks*, 47(4):445–487, 2005.

[5] L. Alfandari. Improved approximation of the soft-capacitated facility location problem. *RAIRO Operations Research*, 41(1):83–93, 2007.

[6] A. Aloisio, V. Izzo, and S. Rampone. Vlsi implementation of greedy-based distributed routing schemes for ad hoc networks. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 11(9):865–872, 2007.

[7] A.C.F. Alvim, C.C. Ribeiro, F. Glover, and D.J. Aloise. A hybrid improvement heuristic for the one-dimensional bin packing problem. *Journal of Heuristics*, 10(2):205–229, 2004.

[8] E. Amaldi, A. Capone, and F. Malucelli. Planning UMTS base station location: Optimization models with power control and algorithms. *Wireless Communications, IEEE Transactions on*, 2(5):939–952, 2003.

[9] J.G. Andrews, A. Ghosh, and R. Muhamed. *Fundamentals of WiMAX: understanding broadband wireless networking*. Prentice Hall PTR, 2007.

[10] S. Anily, J. Bramel, and D. Simchi-Levi. Worst-case analysis of heuristics for the bin packing problem with general cost structures. *Operations research*, 42(2):287–298, 1994.

[11] JP Arabeyre, J. Fearnley, FC Steiger, and W. Teather. The airline crew scheduling problem: A survey. *Transportation Science*, 3(2):140–163, 1969.

[12] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998.

[13] M. Åstrand and J. Suomela. Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks. In *Proceedings of the 22nd ACM symposium on Parallelism in algorithms and architectures*, pages 294–302. ACM, 2010.

[14] J. Aubin. Scheduling ambulances. *Interfaces*, pages 1–10, 1992.

[15] E. Balas and A. Ho. Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study. *Combinatorial Optimization*, pages 37–60, 1980.

[16] N. Bansal, A. Caprara, and M. Sviridenko. Improved approximation algorithms for multidimensional bin packing problems. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 697–708. IEEE, 2006.

[17] N. Bansal, R. Krishnaswamy, and B. Saha. On capacitated set cover problems. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 38–49, 2011.

[18] N. Bansal and K. Pruhs. The geometry of scheduling. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 407–414. IEEE, 2010.

[19] C. Barnhart, C.A. Hane, and P.H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, pages 318–326, 2000.

[20] JE Beasley. An algorithm for set covering problem. *European Journal of Operational Research*, 31(1):85–93, 1987.

[21] JE Beasley. A lagrangian heuristic for set-covering problems. *Naval Research Logistics (NRL)*, 37(1):151–164, 1990.

[22] JE Beasley and P.C. Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94(2):392–404, 1996.

[23] JE Beasley and K. Jörnsten. Enhancing an algorithm for set covering problems. *European Journal of Operational Research*, 58(2):293–300, 1992.

[24] B. Berger, J. Rompel, and P.W. Shor. Efficient nc algorithms for set cover with applications to learning and geometry. *Journal of Computer and System Sciences*, 49(3):454–477, 1994.

[25] P. Berman, M. Karpinski, and A. Lingas. Exact and approximation algorithms for geometric and capacitated set cover problems. *Computing and Combinatorics*, pages 226–234, 2010.

[26] A. Bley. Approximability of unsplittable shortest path routing problems. *Networks*, 54(1):23–46, 2009.

[27] J.A. Bondy and U.S.R. Murty. *Graph theory with applications*, volume 290. MacMillan London, 1976.

[28] N. Bourgeois, B. Escoffier, and V.T. Paschos. Efficient approximation of min set cover by moderately exponential algorithms. *Theoretical Computer Science*, 410(21-23):2184–2195, 2009.

[29] C. Boutevin, M. Gourgand, and S. Norre. Bin packing extensions for solving an industrial line balancing problem. In *Assembly and Task Planning, 2003, Proceedings of the IEEE International Symposium on*, pages 115–121. IEEE, 2003.

[30] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

[31] MJ Brusco, GM Thompson, and LW Jacobs. A morph-based simulated annealing heuristic for a modified bin-packing problem. *Journal of the Operational Research Society*, 48(4):433–439, 1997.

[32] A. Caprara, M. Fischetti, and P. Toth. A heuristic method for the set covering problem. *Operations research*, pages 730–743, 1999.

[33] T. Carnes and D. Shmoys. Primal-dual schema for capacitated covering problems. *Integer Programming and Combinatorial Optimization*, pages 288–302, 2008.

[34] R.D. Carr, L.K. Fleischer, V.J. Leung, and C.A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 106–115. Society for Industrial and Applied Mathematics, 2000.

[35] D. Chakrabarty, E. Grant, and J. Könemann. On column-restricted and priority covering integer programs. *Integer Programming and Combinatorial Optimization*, pages 355–368, 2010.

[36] T.M. Chan, E. Grant, J. Könemann, and M. Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. *Proc. 23rd SODA*, 2012, 2011.

[37] AK Chandra, DS Hirschberg, and CK Wong. Bin packing with geometric constraints in computer network design. *Operations Research*, 26(5):760–772, 1978.

[38] J. Chang, S.Y. Chang, S.P. Hong, Y.H. Min, and M.J. Park. Approximation of a batch consolidation problem. *Networks*, 58(1):12–19, 2011.

[39] YL Chen and YH Chin. The quickest path problem. *Computers & Operations Research*, 17(2):153–161, 1990.

[40] J. Chuzhoy and J.S. Naor. Covering problems with hard capacities. *SIAM Journal on Computing*, 36:498, 2006.

[41] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, pages 233–235, 1979.

[42] E.G. Coffman Jr, MR Garey, and DS Johnson. Approximation algorithms for bin packing: A survey. *Approximation Algorithms for NP-hard Problems*, pages 46–93, 1996.

[43] T.H. Cormen. *Introduction to algorithms*. The MIT press, 2001.

[44] G. Cormode, H. Karloff, and A. Wirth. Set cover algorithms for very large datasets. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 479–488. ACM, 2010.

[45] J.R. Correa and L. Epstein. Bin packing with controllable item sizes. *Information and Computation*, 206(8):1003–1016, 2008.

[46] T.G. Crainic, G. Perboli, M. Pezzuto, and R. Tadei. Computing the asymptotic worst-case of bin packing lower bounds. *European journal of operational research*, 183(3):1295–1303, 2007.

[47] T.G. Crainic, G. Perboli, M. Pezzuto, and R. Tadei. New bin packing fast lower bounds. *Computers & operations research*, 34(11):3439–3457, 2007.

[48] B. Crawford and C. Castro. Integrating lookahead and post processing procedures with aco for solving set partitioning and covering problems. *Artificial Intelligence and Soft Computing–ICAISC 2006*, pages 1082–1090, 2006.

[49] J. Crichigno and B. Barán. Multiobjective multicast routing algorithm for traffic engineering. In *Computer Communications and Networks, 2004. ICCCN 2004. Proceedings. 13th International Conference on*, pages 301–306. IEEE, 2004.

[50] G. Desaulniers, J. Desrosiers, and M.M. Solomon. *Column generation*, volume 5. Springer-Verlag New York Inc, 2005.

[51] Y. Dinitz, N. Garg, and M.X. Goemans. On the single-source unsplittable flow problem. *Combinatorica*, 19(1):17–41, 1999.

[52] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):29–41, 1996.

[53] K.A. Dowsland. Genetic algorithms-a tool for OR. *Journal of the operational Research Society*, 47:550–561, 1996.

[54] R. Duh and M. Fürer. Approximation of k-set cover by semi-local optimization. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 256–264. ACM, 1997.

[55] T. Erlebach and A. Hall. Np-hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow. *Journal of Scheduling*, 7(3):223–241, 2004.

[56] L. Erwu, W. Dongyao, L. Jimin, S. Gang, and J. Shan. Performance evaluation of bandwidth allocation in 802.16j mobile multi-hop relay networks. In *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, pages 939–943. IEEE, 2007.

[57] G. Even, R. Levi, D. Rawitz, B. Schieber, S.M. Shahar, and M. Sviridenko. Algorithms for capacitated rectangle stabbing and lot sizing with joint set-up costs. *ACM Transactions on Algorithms (TALG)*, 4(3):34, 2008.

[58] E. Falkenauer. A hybrid grouping genetic algorithm for bin packing. *Journal of heuristics*, 2(1):5–30, 1996.

[59] E. Falkenauer and A. Delchambre. A genetic algorithm for bin packing and line balancing. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 1186–1192. IEEE, 1992.

[60] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[61] S.P. Fekete and J. Schepers. New classes of fast lower bounds for bin packing problems. *Mathematical programming*, 91(1):11–31, 2001.

[62] T.A. Feo and M.G.C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8(2):67–71, 1989.

[63] M.L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management science*, pages 1–18, 1981.

[64] M.L. Fisher and P. Kedia. Optimal solution of set covering/partitioning problems using dual heuristics. *Management Science*, 36:674–688, 1990.

[65] K. Fleszar and K.S. Hindi. New heuristics for one-dimensional bin-packing. *Computers & operations research*, 29(7):821–839, 2002.

[66] F.V. Fomin, F. Grandoni, and D. Kratsch. Some new techniques in design and analysis of exact (exponential) algorithms. *Bulletin-European Association for Theoretical Computer Science*, 87:47, 2005.

[67] D.R. Ford and D.R. Fulkerson. *Flows in networks*. Princeton university press, Princeton, N.J., 1962.

[68] D.K. Friesen and M.A. Langston. Variable sized bin packing. *SIAM journal on computing*, 15(1):222–230, 1986.

[69] I.K. Fu, W.H. Sheen, and F.C. Ren. Deployment and radio resource reuse in IEEE 802.16j multi-hop relay network in Manhattan-like environment. In *Information, Communications & Signal Processing, 2007 6th International Conference on*, pages 1–5. IEEE, 2007.

[70] DR Fulkerson, G.L. Nemhauser, and L.E. Trotter. Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triple systems. *Approaches to Integer Programming*, pages 72–81, 1974.

[71] T. Gabriel Crainic, G. Perboli, W. Rei, and R. Tadei. Efficient lower bounds and heuristics for the variable cost and size bin packing problem. *Computers & Operations Research*, 38:1474–1482, 2011.

[72] R. Gandhi, E. Halperin, S. Khuller, G. Kortsarz, and A. Srinivasan. An improved approximation algorithm for vertex cover with hard capacities. *Journal of Computer and System Sciences*, 72(1):16–33, 2006.

[73] Y. Ge, S. Wen, Y.H. Ang, and Y.C. Liang. Optimal relay selection in IEEE 802.16j multihop relay vehicular networks. *Vehicular Technology, IEEE Transactions on*, 59(5):2198–2206, 2010.

[74] V. Genc, S. Murphy, and J. Murphy. Analysis of transparent mode IEEE 802.16j system performance with varying numbers of relays and associated transmit power. In *Wireless Communications and Networking Conference, 2009. WCNC 2009. IEEE*, pages 1–6. IEEE, 2009.

[75] V. Genc, S. Murphy, Y. Yu, and J. Murphy. IEEE 802.16j relay-based wireless access networks: an overview. *Wireless Communications, IEEE*, 15(5):56–63, 2008.

[76] P.C. Gilmore and R.E. Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961.

[77] P.C. Gilmore and R.E. Gomory. A linear programming approach to the cutting stock problem-part II. *Operations research*, 11(6):863–888, 1963.

[78] Fred Glover and Claude McMillan. The general employee scheduling problem. an integration of ms and ai. *Computers & OR*, 13(5):563–573, 1986.

[79] O. Goldschmidt, D.S. Hochbaum, and G. Yu. A modified greedy heuristic for the set covering problem with improved worst case bound. *Information Processing Letters*, 48(6):305–310, 1993.

[80] F. Grandoni. A note on the complexity of minimum dominating set. *Journal of Discrete Algorithms*, 4(2):209–214, 2006.

[81] T. Grossman and A. Wool. Computational experience with approximation algorithms for the set covering problem. *European Journal of Operational Research*, 101(1):81–92, 1997.

[82] P. Hansen and N. Mladenovic. Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449–467, 2001.

[83] F. Harche and G.L. Thompson. The column subtraction algorithm: an exact method for solving weighted set covering, packing and partitioning problems. *Computers & operations research*, 21(6):689–705, 1994.

[84] M. Held and R.M. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, pages 1138–1162, 1970.

[85] M. Held and R.M. Karp. The traveling-salesman problem and minimum spanning trees: Part ii. *Mathematical programming*, 1(1):6–25, 1971.

[86] D.S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11:555, 1982.

[87] E. Hopper and BCH Turton. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research*, 128(1):34–57, 2001.

[88] W.C. Huang, C.Y. Kao, and J.T. Horng. A genetic algorithm approach for set covering problems. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 569–574. IEEE, 1994.

[89] R. Hubscher and F. Glover. Applying tabu search with influential diversification to multiprocessor scheduling. *Computers & operations research*, 21(8):877–884, 1994.

[90] F.K. Hwang and D.S. Richards. Steiner tree problems. *Networks*, 22(1):55–89, 1992.

[91] S.M. Hwang, C.Y. Kao, and J.T. Horng. On solving rectangle bin packing problems using genetic algorithms. In *Systems, Man, and Cybernetics, 1994.'Humans, Information and Technology'., 1994 IEEE International Conference on*, volume 2, pages 1583–1590. IEEE, 1994.

[92] IBM ILOG. *CPLEX, Version 12.1.0*. Website http://www.ilog.com/products/cplex/, 2004.

[93] H. Iima and T. Yakawa. A new design of genetic algorithm for bin packing. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 2, pages 1044–1049. IEEE, 2003.

[94] L.W. Jacobs and M.J. Brusco. A simulated annealing based heuristic for the set-covering problem. In *Working paper, Operations Management and Information Systems Department, Northern Illinois University, Dekalb, IL*, 1993.

[95] D.S. Johnson. Approximation algorithms for combinatorial problems*. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

[96] D.S. Johnson and M.R. Garey. Computers and intractability: A guide to the theory of np-completeness. *Freeman&Co, San Francisco, CA*, 1979.

[97] C.Y. Kao and F.T. Lin. A stochastic approach for the one-dimensional bin-packing problems. In *Systems, Man and Cybernetics, 1992., IEEE International Conference on*, pages 1545–1551. IEEE, 1992.

[98] N. Karmarkar. An interior point approach to np-complete problems — part i. *Contemporary Mathematics*, 114:297–308, 1990.

[99] N. Karmarkar, M.G.C. Resende, and KG Ramakrishnan. An interior point algorithm to solve computationally difficult set covering problems. *Mathematical Programming*, 52(1):597–618, 1991.

[100] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE, 1995.

[101] S. M. Khaled, R. Benkoczi, and Y. Chen. Mesh network deployment to ensure global reachability. In *Integrated Enabling Technologies, 2012. IET'12. 1st International Workshop on*, pages 207–213, Vancouver, Canada, June 2012. IEEE.

[102] J. M. Kleinberg. *Approximation problems for disjoint path problems*. PhD thesis, MIT, Cambridge, MA, 1996.

[103] J.M. Kleinberg. Single-source unsplittable flow. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 68–77. IEEE, 1996.

[104] S. Kolliopoulos and C. Stein. Experimental evaluation of approximation algorithms for single-source unsplittable flow. *Integer Programming and Combinatorial Optimization*, pages 328–344, 1999.

[105] S.G. Kolliopoulos and C. Stein. Improved approximation algorithms for unsplittable flow problems. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 426–436. IEEE, 1997.

[106] S.G. Kolliopoulos and C. Stein. Approximation algorithms for single-source unsplittable flow. *SIAM Journal on Computing*, 31:919, 2001.

[107] P. Kolman and C. Scheideler. Improved bounds for the unsplittable flow problem. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 184–193. Society for Industrial and Applied Mathematics, 2002.

[108] B. Kwon, Y. Chang, and J.A. Copeland. A network entry protocol and an OFDMA symbol allocation scheme for non-transparent relay stations in IEEE 802.16j MMR networks. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1–6. IEEE, 2008.

[109] K. Lee and J. Jang. Reduction of relay overhead in IEEE 802.16j mobile multi-hop relay (MMR) networks. In *Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE*, pages 1–2. IEEE, 2009.

[110] J.K. Lenstra, D.B. Shmoys, and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical programming*, 46(1):259–271, 1990.

[111] L. Lessing, I. Dumitrescu, and T. Stützle. A comparison between aco algorithms for the set covering problem. *Ant Colony Optimization and Swarm Intelligence*, pages 105–122, 2004.

[112] J. Levine and F. Ducatelle. Ant colony optimization and local search for bin packing and cutting stock problems. *Journal of the Operational Research Society*, 55(7):705–716, 2004.

[113] XY Li, YP Aneja, and F. Baki. An ant colony optimization metaheuristic for single-path multicommodity network flow problems&star. *Journal of the Operational Research Society*, 61(9):1340–1355, 2009.

[114] G. E. Liepins, M. R. Hilliard, M. Palmer, and M. Monow. Greedy genetics. In *Genetic Algorithms, Cambridge, MA, USA, July 1987. 2nd Internationd Conference on*, pages 90–99. Lawrence Erlbaum Associates, 1987.

[115] G. E. Liepins, M. R. Hilliard, J. Richardson, and M. Palmer. Genetic algorithms applications to set covercing and traveling salesman problems. In *Operations Research and Artificial Intelligence: The Integration of Problem-Solving Stmtegies*, pages 29–57. Kluwer Academic Publishers, Norwell, Massachusetts, 1990.

[116] B. Lin, P.H. Ho, L.L. Xie, and X. Shen. Optimal relay station placement in IEEE 802.16j networks. In *Proceedings of the 2007 international conference on Wireless communications and mobile computing*, pages 25–30. ACM, 2007.

[117] B. Lin, P.H. Ho, L.L. Xie, and X. Shen. Relay station placement in IEEE 802.16j dual-relay MMR networks. In *Communications, 2008. ICC'08. IEEE International Conference on*, pages 3437–3441. IEEE, 2008.

[118] P. Lin, H. Ngo, C.M. Qiao, X. Wang, T. Wang, and D.Y. Qian. Minimum cost wireless broadband overlay network planning. In *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, pages 228–236. IEEE Computer Society, 2006.

[119] S.J. Lin, W.H. Sheen, I.K. Fu, and C. Huang. Resource scheduling with directional antennas for multi-hop relay networks in Manhattan-like environment. In *Mobile WiMAX Symposium, 2007. IEEE*, pages 108–113. IEEE, 2007.

[120] V.I. Litvinenko, JA Burgher, A.A. Tkachuk, and V.J. Gnatjuk. The application of the distributed genetic algorithm to the decision of the packing in containers problem. In *Artificial Intelligence Systems, 2002.(ICAIS 2002). 2002 IEEE International Conference on*, pages 386–390. IEEE, 2002.

[121] DS Liu, KC Tan, SY Huang, CK Goh, and WK Ho. On solving multiobjective bin packing problems using evolutionary particle swarm optimization. *European Journal of Operational Research*, 190(2):357–382, 2008.

[122] A. Lodi, S. Martello, and D. Vigo. Heuristic algorithms for the three-dimensional bin packing problem. *European Journal of Operational Research*, 141(2):410–420, 2002.

[123] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete mathematics*, 13(4):383–390, 1975.

[124] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[125] H. Masri, S. Krichen, and A. Guitouni. An ant colony optimization metaheuristic for solving bi-objective multi-sources multicommodity communication flow problem. In *Wireless and Mobile Networking Conference (WMNC), 2011 4th Joint IFIP*, pages 1–8. IEEE, 2011.

[126] N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[127] Baldi MM, Crainic TG, Perboli G, and Tadei R. *The generalized bin packing problem*. CIRRELT-2010-21 Universit de Montral, Montral, QC, Canada URL: https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2011-39.pdf(Accessed on 05-Feb-2012), 2010.

[128] G.L. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization*, volume 18. Wiley New York, 1988.

[129] D. Niyato, E. Hossain, D.I. Kim, and Z. Han. Relay-centric radio resource management and network planning in IEEE 802.16j mobile multihop relay networks. *Wireless Communications, IEEE Transactions on*, 8(12):6115–6125, 2009.

[130] IEEE 802: Overview and Architecture [ONLINE]. *IEEE Standards Association*. URL: http://standards.ieee.org/about/get/802/802.html (Accessed on 28-Jan-2012).

[131] G.S. Paschos, P. Mannersalo, and T.M. Bohnert. Cell capacity for IEEE 802.16 coverage extension. In *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, pages 933–937. IEEE, 2008.

[132] D. Peleg, G. Schechtman, and A. Wool. Approximating bounded 0-1 integer linear programs. In *Theory and Computing Systems, 1993., Proceedings of the 2nd Israel Symposium on the*, pages 69–77. IEEE, 1993.

[133] C. Peng, Y. Tan, and L.T. Yang. New algorithms for the minimum-cost single-source unsplittable flow problem. In *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, volume 1, pages 136–141. IEEE, 2007.

[134] S.W. Peters and R.W. Heath. The future of WiMAX: Multihop relaying with IEEE 802.16j. *Communications Magazine, IEEE*, 47(1):104–111, 2009.

[135] C. Pimpawat and N. Chaiyaratana. Using a co-operative co-evolutionary genetic algorithm to solve a three-dimensional container loading problem. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 2, pages 1197–1204. IEEE, 2001.

[136] M. Pióro, D. Medhi, and ScienceDirect (Online service). *Routing, flow, and capacity design in communication and computer networks*. Elsevier/Morgan Kaufmann, 2004.

[137] C. Reeves. Hybrid genetic algorithms for bin-packing and related problems. *Annals of Operations Research*, 63(3):371–396, 1996.

[138] Z. Ren, Z. Feng, L. Ke, and H. Chang. A fast and efficient ant colony optimization approach for the set covering problem. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 1839–1844. IEEE, 2008.

[139] C. Revelle, D. Marks, and J.C. Liebman. An analysis of private and public sector location models. *Management Science*, pages 692–707, 1970.

[140] A. Scholl, R. Klein, and C. Jürgens. BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers & Operations Research*, 24(7):627–645, 1997.

[141] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons Inc, 1998.

[142] A. Schrijver. *Combinatorial optimization*. Springer, 2003.

[143] P. Schwerin and G. Waescher. A new lower bound for the bin-packing problem and its integration into MTP. *Pesquisa Operacional*, 19:111–129, 1999.

[144] S. Sen. Minimal cost set covering using probabilistic methods. In *Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing: states of the art and practice*, pages 157–164. ACM, 1993.

[145] Y. Shigehiro, S. Koshiyama, and T. Masuda. Stochastic tabu search for rectangle packing. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 4, pages 2753–2758. IEEE, 2001.

[146] M. Skutella. Approximating the single source unsplittable min-cost flow problem. *Mathematical Programming*, 91(3):493–514, 2002.

[147] P. Slavík. A tight analysis of the greedy algorithm for set cover. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 435–441. ACM, 1996.

[148] A. Srinivasan. Improved approximations of packing and covering problems. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 268–276. ACM, 1995.

[149] K. Sundaresan and S. Rangarajan. On exploiting diversity and spatial reuse in relay-enabled wireless networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 13–22. ACM, 2008.

[150] Z. Tao, A. Li, K.H. Teo, and J. Zhang. Frame structure design for IEEE 802.16j mobile multihop relay (MMR) networks. In *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*, pages 4301–4306. IEEE, 2007.

[151] Z. Tao, K.H. Teo, and J. Zhang. Aggregation and concatenation in IEEE 802.16j mobile multihop relay (MMR) networks. In *Mobile WiMAX Symposium, 2007. IEEE*, pages 85–90. IEEE, 2007.

[152] JM Valério de Carvalho. Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, 86:629–659, 1999.

[153] JM Valério de Carvalho. LP models for bin packing and cutting stock problems. *European Journal of Operational Research*, 141(2):253–273, 2002.

[154] H. Van De Vel and S. Shijie. An application of the bin-packing technique to job scheduling on uniform processors. *Journal of the Operational Research Society*, 42(2):169–172, 1991.

[155] Johan M. M. van Rooij and Hans L. Bodlaender. Design by measure and conquer, a faster exact algorithm for dominating set. *Computing and Research Repository*, abs/0802.2827, 2008.

[156] F.J. Vasko. An efficient heuristic for large set covering problems. *Naval Research Logistics Quarterly*, 31(1):163–171, 1984.

[157] E. Visotsky, J. Bae, R. Peterson, R. Berry, and M.L. Honig. On the uplink capacity of an 802.16j system. In *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE*, pages 2657–2662. IEEE, 2008.

[158] K. Walkowiak. New algorithms for the unsplittable flow problem. *Computational Science and Its Applications-ICCSA 2006*, pages 1101–1110, 2006.

[159] L.A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[160] M. Yagiura, M. Kishida, and T. Ibaraki. A 3-flip neighborhood local search for the set covering problem. *European Journal of Operational Research*, 172(2):472–499, 2006.

[161] L.H.W. Yeung and W.K.S. Tang. A hybrid genetic approach for container loading in logistics industry. *Industrial Electronics, IEEE Transactions on*, 52(2):617–627, 2005.

[162] Y. Yu, S. Murphy, and L. Murphy. A clustering approach to planning base station and relay station locations in IEEE 802.16j multi-hop relay networks. In *Communications, 2008. ICC'08. IEEE International Conference on*, pages 2586–2591. IEEE, 2008.

[163] Y. Yu, S. Murphy, and L. Murphy. Planning base station and relay station locations in IEEE 802.16j multi-hop relay networks. In *IEEE Consumer Communications Networking Conference (CCNC) 2008*, pages 922–926. IEEE, 2008.

[164] J. Zhang, S. Feng, W. Ye, and H. Zhuang. MAC Performance Evaluation of IEEE 802.16j. In *Information Science and Engineering, 2008. ISISE'08. International Symposium on*, volume 1, pages 421–425. IEEE, 2008.