



TAMPERE UNIVERSITY OF TECHNOLOGY

**Qiao Wang**

**On the Optimal Assisted Rate Allocation in N-Tier Multi-RAT  
Heterogeneous Networks**

Master of Science Thesis

Examiner: Dr. Dmitri Moltchanov,  
Prof. Yevgeni Koucheryavy  
Examiner and topic approved by the  
Faculty Council of the Faculty of  
Computing and Electrical Engineering  
on 22 August 2014

# ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY

Master's Degree Programme in Information Technology

**WANG QIAO: On the Optimal Assisted Rate Allocation in N-Tier Multi-RAT Heterogeneous Networks**

Master of Science Thesis, 53 pages

August 2014

Major: Communication Systems and Networks

Examiner: Dmitri Moltchanov

Keywords: NFP, radio access technology, heterogeneous networks, LTE, WiFi, fair networks, Max-min fairness, AMPL

Telecommunication industry is facing a brand new era nowadays. As the number of mobile devices increases dramatically, people need to figure out an efficient way to allocate wireless resources and under this background, the resource allocation in heterogeneous network has drawn much attention. Heterogeneous network is the network combines large cellular cells with smaller ones by various radio access technologies (RATs). Subscribers or users may be located in random places within the coverage of a heterogeneous network. So how to fairly provide wireless access service for the users is the main subject of resource allocation.

This thesis is to achieve the above objective. More specifically, our research formulated heterogeneous networks resource allocation problem as a *network flow problem*, where each subscriber is considered as a traffic demand and heterogeneous network performs as a shared link that offers traffic for them. The main method to realize fairness is through *max-min fairness* (MMF) criterion and to make the results more convincing, 50 topologies have been established although only 5 of them are chosen for illustration. The algorithm adopted is based on *non-blocking test*. With these topologies, the author carefully built the mathematical model in AMPL software, which is the critical implementation tool for this thesis.

After accurate formulation and modeling, the author has concluded that N-tier heterogeneous network resource allocation problem can be efficiently solved by considering it as a network flow problem.

## PREFACE

Starting from November, 2013 to May, 2014, I finally complete this master's thesis. Undoubtedly, I have met many tricky problems. At first, learning the relevant materials was somehow exhausting and there was a time I found it was much more difficult than I expected. Then, when I totally understood what I was going to do, the problem I encountered was how to implement the algorithms. But, anyway, I have successfully dealt with them and obtained valuable experience.

During my thesis process, my supervisor, Dmitri Moltchanov, has offered me much help. He kept encouraging and motivating me. Besides, he set meeting once a week with me to discuss my thesis progress and to see if I had some problems no matter how busy he was. Here I would like to thank him for his guidance and encouragement and show my great respect to him.

Finally, I want to thank my parents for supporting me to study abroad and their spiritual support as well. Also, thank my friends for their company in both studying and usual life.

Qiao Wang  
2014-08-22  
Tampere

# CONTENTS

1. Introduction . . . . .	1
2. Network Flow Problem . . . . .	4
2.1 A Network Flow Example . . . . .	4
2.2 Notations of NFP . . . . .	8
2.3 The Appropriateness of NFP . . . . .	10
2.4 Fair Networks . . . . .	10
2.4.1 Notion of Fair Networks . . . . .	10
2.4.2 Max-Min Fairness for Fixed Paths . . . . .	12
3. Formulation and Solution of Heterogeneous Network Systems . . . . .	15
3.1 NFP Formulation of Our Problem . . . . .	15
3.1.1 System Description . . . . .	15
3.1.2 Formulation . . . . .	18
3.2 Solution Algorithm . . . . .	22
3.3 AMPL As a Solving Tool . . . . .	26
3.3.1 Background of Mathematical Programming . . . . .	26
3.3.2 Introduction to AMPL Development Environment and Language Basics . . . . .	27
3.3.3 AMPL Application in Solving 3-tier Heterogeneous Network Prob- lem . . . . .	31
4. Numerical Results of Heterogeneous Network Problem . . . . .	41
4.1 A Simple Situation . . . . .	41
4.2 Numerical Results of More Realistic Cases . . . . .	43
4.3 Comparison of Three Schemes . . . . .	50
5. Conclusion . . . . .	52
References . . . . .	54

## TERMS AND DEFINITIONS

NFP	Network flow problem
RAT	Radio access technology
HetNet	Heterogeneous network
BS	Base station
AMPL	AMPL modeling language

# 1. INTRODUCTION

Living in an information era the human beings are connected by cellular communication network. The global cellular communication network, one of the most essential achievements in telecommunication field, has been in the process of "paradigm shift" because of the increasing number of base stations (BSs) each year [1]. It has been studied that by 2015, the number of BSs will reach 50 million and in the near future it will outnumber the subscribers [2] [3]. Consequently, for each user there will be at least one base station for use. The reason of this expansion is that consumers are requiring a faster data rate and it is impossible to satisfy their requests by simply adding spectrum [1]. Thus, small BSs (micro, pico, femto, etc.) emerged and have become increasingly feasible from both finance and technology perspective. In fact, the escalation of BSs is via adding small BSs into the existing network [1]. Therefore, it is clear to see that future networks are anticipated to support many different scenarios and applications by (i) employing network densification i.e., using small BSs and (ii) a tighter combination between various radio access technologies (RATs) [4]. As a result, heterogeneous networks cropped up and has been investigated and developed.

Heterogeneous network (HetNet) (see Figure 1) is a hierarchical deployment that combines large (macro) cells with small ones (micro, pico, femto) using different RATs (3G, LTE, WiFi) [19]. The objective is to provide "ubiquitous coverage and connectivity [4]" in order to satisfy the burgeoning requirement for wireless data. So far, it is necessary to have a brief explanation for small cells and RATs as well. Firstly, small cells, generally including femtocells, picocells, and microcells, are low-powered radio access and operator controlled nodes which operate in licensed and unlicensed spectrum with typically a range of 10 meters to 1 or 2 kilometres [5]. Compared with macro cells which may have a range of 10 kilometres or more, they are "small" and important to managing LTE Advanced spectrum. Secondly, Long-Term Evolution (LTE), a standard developed by 3rd Generation Partnership Project (3GPP) for wireless communication of high-speed data rate, has been accessible in many European countries and also Asian countries like China, Japan and Korea [13]. Its advanced version, named LTE-A with focus on higher capacity, has also been developed in order to obtain a higher bitrate in a cost-efficient way [15]. Thirdly, WiFi is basically a wireless networking technology which allows mobile devices such

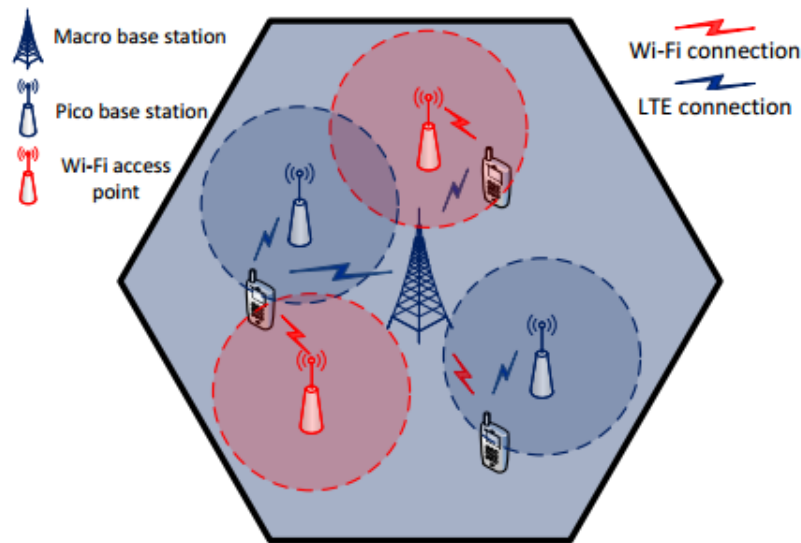


Figure 1.1: Simplified deployment example of a heterogeneous network

as cellphone and laptop to connect to the Internet. "The Wi-Fi Alliance defines Wi-Fi as 'any wireless local area network (WLAN) products that are based on the Institute of Electrical and Electronics Engineers' (IEEE) 802.11 standards' [6]".

It is commonly believed that through integrating multiple layers of multi-radio small cells, the next-generation HetNet will dramatically enhance the network capacity and quality of experience (QoE). Particularly, a centralized control node which resides in macro cell will intelligently associate users with the network infrastructure and manage their data flow. This is what we call as assisted network selection and it has greatly attracted the industry's interest. [14][17]

Despite the significance and inevitability of assisted network, the corresponding assisted data rate allocation problem has not been studied sufficiently in the past academic investigations. For example, in [7] and [8] stochastic geometry models were employed and the results were reasonable "highlighting the achievable upper and lower bound capacity of heterogeneous system [4]". Indeed, these conclusions are of great importance for the understanding of next-generation N-tier HetNet but the problem has not been comprehensively solved.

To fill the gap, this thesis treated the indicated problem as a network flow problem where each user was considered as a source of traffic demand and each tier of N-tier HetNet was represented as a shared link providing traffic for a fraction of these demands. In general, we have built enough number of HetNet topologies and each of them have 3 layers together with 20 subscribers. These users are considered as demands and randomly distributed in the coverage of different RATs i.e., WiFi,

micro-LTE or macro-LTE. Moreover, we reformulated the optimization algorithm that is proposed for solving Max-Min Fairness problem, which will be introduced in detail in the following sections. Furthermore, as will be illustrated, the resource allocation in N-tier multi-RAT networks is an instance of a linear programming problem, which is why we used AMPL mathematical model language to implement it. Our result demonstrated that the algorithm worked perfectly on the established model and each subscriber obtained fair network flow/demand volume. Another objective of this thesis is to attract the attention of more researchers and encourage them to study the future Heterogeneous networks by modelling it into a network flow problem.

The remainder of this paper is divided into four sections. In the second section, network flow problem will be introduced in detail: what it is, why it is appropriate and how it is notated. Also, the notion and algorithm of fair networks are explained. The third chapter will contain the actual problem of our research and the solution algorithm and the corresponding solving tool. Chapter four is for the numerical results where we will show and analyse the results of some particular scenarios. Lastly, the conclusion section will include a comprehensive and elaborate summary.

## 2. NETWORK FLOW PROBLEM

This chapter is fairly vital because it covers the primary principle behind this thesis and illustrates the notation method used in the paper work. In this chapter, a very important algorithmic problem, network flow problem (NFP), will be illustrated. NFP is critical because it can be used to express many kinds of problems. Four sections are included in this chapter: we will start with an example of NFP; then introduce a brief and efficient notation; discuss a small summary about the appropriateness of network flow problem; finally, illustrate in detail a famous network problem—fair networks.

### 2.1 A Network Flow Example

Here is a simple network example (see book [9] for more details) where three nodes are connected with each other, i.e., the network topology will look like a triangle (see Figure 2.1 [9]). In this case, nodes can be either routers in the Internet or switches in telephone networks or even digital cross-connects in SONET network [9]. We will use *node* throughout this paper since it is a generic term to identify various routing or switching devices in networks. Another term is *demand volume* [9] representing "either the traffic volume (as in the Internet or the telephone network) or the required bandwidth (as in SONET) between a pair of nodes, depending on the considered type of network [9]". Such a pair of nodes is called demand [9].

To keep this example simple enough, it is assumed that the demands and the link between them are bi-directional, which in general can be directed. Now, suppose that between node 1 and 2, node 1 and 3, node 3 and 2 the demand volume is 5, 7 and 8 respectively and we use  $h$  to identify these volumes:

$$h_{12} = 5, h_{13} = 7, h_{23} = 8 \quad (2.1)$$

There are two paths in this topology for the demand volume of each pair of nodes to be routed. For instance, the demand pair with node 2 and node 3 (denoted as  $\langle 2,3 \rangle$ ) has route 2-3 and an alternate route 2-1-3 via node 1 for routing its demand volume. The amount of demand volume that will be routed on each path will primarily rely on the objective function which will be discussed more in the following. Thus, if  $x$  can be used to identify the *path-flow variables* (flow variables

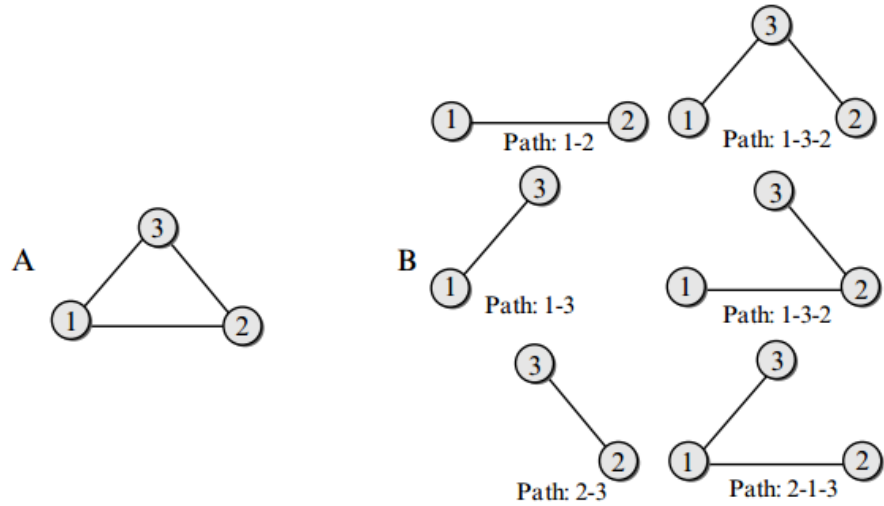


Figure 2.1: (A) Three-Node Network Example and (B) All Possible Paths for the Three-Node Example [9]

for simplification), the equations for pair  $\langle 2,3 \rangle$  below will be wrote:

$$x_{23} + x_{213} = h_{23} \quad (8) \quad (2.2)$$

The subscripts of variable  $x$  here are to identify the paths, path 2-3 and 2-1-3. Similarly, for pairs  $\langle 1,2 \rangle$  and  $\langle 1,3 \rangle$  semblable equations can also be concluded:

$$x_{12} + x_{132} = h_{12} \quad (5), \quad x_{13} + x_{123} = h_{13} \quad (7) \quad (2.3)$$

It is necessary to mention that path-flows are non-negative.

Besides the above two items, another factor we need to consider is *link capacity*. In this example, links are denoted by such as 1-2, 1-3 and 2-3 and the associated capacity are  $c_{12}$ ,  $c_{13}$  and  $c_{23}$  respectively with the subscripts denoting the end nodes of a particular link. Note that the units of demand volume and link capacity need to be consistent, i.e., if we are using packet per second (pps) as the unit of demand volume, then link capacity should also be expressed in the same unit or units that can be transfered to pps [9]. Another point worth mentioning is that link connects two nodes directly while the demand volume can be between any pair of nodes [9].

The next task is to find out which flows might use different links. Apparently, flow variables  $x_{12}$ ,  $x_{123}$  and  $x_{213}$  use link 1-2 with a capacity  $c_{12}$  and considering the common knowledge that in any communication network link load cannot exceed link capacity, the following inequality of link 1-2 can be written:

$$x_{12} + x_{123} + x_{213} \leq c_{12} \quad (2.4)$$

For other two links 1-3 and 2-3, similarly:

$$x_{13} + x_{132} + x_{213} \leq c_{13} \quad x_{23} + x_{132} + x_{123} \leq c_{23} \quad (2.5)$$

Given the values of link capacity, for example,  $c_{12} = c_{13} = 10$ ,  $c_{23} = 15$  and together with the equations and inequalities (constraints) concluded above, we can obtain:

$$x_{12} + x_{132} = 5$$

$$x_{13} + x_{123} = 7$$

$$x_{23} + x_{213} = 8$$

$$x_{12} + x_{123} + x_{213} \leq 10$$

$$x_{13} + x_{132} + x_{213} \leq 10$$

$$x_{23} + x_{132} + x_{123} \leq 15$$

$$x_{12}, x_{13}, x_{23}, x_{123}, x_{132}, x_{213} \geq 0.$$

The above system referred to as 2.1a, in fact, has multiple *feasible solutions* [9]. However, which of them is of best interest? To solve this, we need to figure out the essential part in terms of the goal of network design problem, which in the context of mathematical representation is known as *objective function* [9].

Now assume that 1 is the cost of routing one unit of flow on every link and the ultimate objective is to minimize the total routing cost, which is:

$$O = x_{12} + x_{13} + x_{23} + 2x_{132} + 2x_{123} + 2x_{213} \quad (2.1b)$$

The coefficients of paths like 1-2-3 are 2 because they consist of two links and it costs twice to route on a two-link path compared with one-link path [9].

Therefore, eventually our goal is to minimize the objective function (2.1b) subject to the constraints in (2.1a) [9]. For completeness, we can write the problem discussed so far as problem (2.1):

*minimize*

$$O = x_{12} + x_{13} + x_{23} + 2x_{132} + 2x_{123} + 2x_{213}$$

*subject to:*

$$x_{12} + x_{132} = 5$$

$$x_{13} + x_{123} = 7$$

$$x_{23} + x_{213} = 8$$

$$x_{12} + x_{123} + x_{213} \leq 10$$

$$x_{13} + x_{132} + x_{213} \leq 10$$

$$x_{23} + x_{132} + x_{123} \leq 15$$

$$x_{12}, x_{13}, x_{23}, x_{123}, x_{132}, x_{213} \geq 0.$$

Problems like (2.1) is a particular type of "*multi-commodity network flow problem* [9]" since there are more than one demand which need to be routed simultaneously and compete for network resources like link capacity in this situation. As seen in many optimization issues, this expression is well known as **linear programming problem** due to the linearity of the constraints and the objective function.

What is still needed in problem (2.1) is to find out the optimum solution i.e., feasible values of  $x$ . Obviously, the optimum solution is fairly easy to find without any professional or specific tools; simply route everything on the direct path since the cost is twice routing on a two-link path. As a result, the final solution is:

$$x_{12} = 5 \quad x_{13} = 7 \quad x_{23} = 8,$$

while other flow variables have a value of 0. The minimum routing cost  $O$  is  $O = 20$ . This solution is optimal and feasible as well since it satisfies all the constraints. Moreover, this optimal solution is unique in this case.

However, it is not always so easy to solve. A small variation can be made to the above problem [9]. Suppose that the routing cost of a unit of flow is twice as expensive to route on a direct path compared to a multiple-link path. The new objective function is:

$$O^* = 2x_{12} + 2x_{13} + 2x_{23} + x_{132} + x_{123} + x_{213}.$$

Apparently, we can use the similar method in the previous example that simply route all the traffic through indirect paths. Unfortunately, the capacity constraints will not be met. But this does not mean there is no suitable values of  $x$ . Note that in the revised problem, the objective function is the only thing that is changed, not the constraints. Thus, the optimal solution obtained in problem (2.1) is still a feasible solution but not the optimal one. In fact, the optimal solution for the revised problem is somewhere between the former solution and the situation when all demand volumes are routed through the cheaper path (multiple-link path in the new problem). Here we will only give the results of it without showing how to solve:

$$x_{12} = 0, x_{13} = 1, x_{23} = 4, x_{132} = 5, x_{123} = 6, x_{213} = 4$$

and the optimal solution is  $O^* = 25$ .

So far, some very basic knowledge about network flow problem has been discussed. In order to understand multi-commodity network problem better, two important lessons [9] can be learned from the above discussion: one is that changing the objective function can influence the optimal solution and the method of finding it; the other is that the goal of a particular network should be carefully considered otherwise the final optimal solution can be meaningless.

## 2.2 Notations of NFP

In this section, a different, brief and more efficient notation will be introduced (see pp.45, in [9]). The introduction is fairly necessary since in the practical problem considered in the rest of this paper will utilize this new notation. In section 2.1, we have used a notation referred to as "*node-identifier-based notation* [9]" in three-node network but it has several drawbacks. Firstly, not all nodes will make a pair with the other nodes, i.e., some pairs may have no demand. Secondly, many intermediate nodes can be contained in one path. Lastly, the length of the indices of flow variables will be different. To illustrate, imagine we have a network with say 100 nodes among which there is no demand between node 4 and node 10 and other pairs. In this case, it is necessary to indicate in the model representation that these pairs do not have any demand requirement. Using "*node-identifier-based notation* [9]", these pairs need to be explicitly listed, for instance, we have demand  $h_{mn}$  except  $h_{410}$  ( $\langle m,n \rangle = \langle 4,10 \rangle$ ) and so on. Another problem is about the link representation. For example, if there is no connection between node 3 and node 7, then not only is the link capacity  $c_{37} = 0$  but the link 3-7 does not need to be represented at all. In addition, as the network gets more complicated, there will be many routes between two nodes and each path will probably have multiple hops.

To avoid these inconveniences, this section will introduce a new notation, referred to as "*link-demand-path-identifier-based notation* [9]". It has some obvious advantages [9]:

- compact and only necessary objects are included
- more convenient to capture and formulate NFP
- allows to make algebraical manipulations on particular problems

In the following content, it will explain how this notation works. First of all, the demand representation. Basically, only the demand pairs that have non-negative demand volume are assigned labels from 1 to the total number of such demands [9]. Thus those nodes with no demand are not listed. Still consider the network example in section 2.1; if we reformulate it using the above notation, the demand can be expressed as:

demand pair  $\langle 1, 2 \rangle \iff 1$

demand pair  $\langle 1, 3 \rangle \iff 2$

demand pair  $\langle 2, 3 \rangle \iff 3$ .

Similarly, links existing in the network can be shown with indices from 1 to the total number:

link 1-2  $\iff 1$

link 1-3  $\iff 2$

link 2-3  $\iff 3$ .

In general, we will use  $D$  and  $d$  to denote the total number of demands and labels of them respectively;  $E$  and  $e$  for the number of links and their labels. For example, still the three-node example,  $D = 3$ ,  $d = 1, 2, 3$ ;  $E = 3$ ,  $e = 1, 2, 3$ .

With the above knowledge, we will have the following mapping for demand volumes and link capacities in problem in section 2.1:

$h_{12} \iff h_1$        $h_{13} \iff h_2$        $h_{23} \iff h_3$

$c_{12} \iff c_1$        $c_{13} \iff c_2$        $c_{23} \iff c_3$ .

After successfully transferring the demand pairs and the links to the new notation, the identifiers for paths need to be discussed. The principle is simple. Now that we have demand pair identifier, this can be used as the first subscript of a path variable and then the second subscript will be the label of the path for that individual demand. To illustrate,  $P_d$  can be noted as the total number of paths for demand  $d$  and the paths are indexed with  $p$  which is numbered from 1 to the total number of paths for that demand. For instance, demand  $\langle 2, 3 \rangle$  with label  $d = 3$  has  $P_3 = 2$  candidate paths: 2-3 and 2-1-3 and  $p = 1, 2$  labeling these two paths respectively. So far, we can re-write the flow variables like this:

$x_{12} \iff x_{11}$ ,       $x_{132} \iff x_{12}$

$x_{13} \iff x_{21}$ ,       $x_{123} \iff x_{22}$

$x_{23} \iff x_{31}$ ,       $x_{213} \iff x_{32}$ .

Hence, with the above background, the linear programming problem in section 2.1 can be reformulate using the new notation:

**minimize**

$$O = x_{11} + x_{21} + x_{31} + 2x_{12} + 2x_{22} + 2x_{32}$$

**subject to:**

$$x_{11} + x_{12} = h_1$$

$$x_{21} + x_{22} = h_2$$

$$x_{31} + x_{32} = h_3$$

$$x_{11} + x_{22} + x_{32} \leq c_1$$

$$x_{21} + x_{12} + x_{32} \leq c_2$$

$$x_{31} + x_{12} + x_{22} \leq c_3$$

$$x_{11}, x_{21}, x_{31}, x_{22}, x_{12}, x_{32} \geq 0.$$

It can be noted that both formulations represent the same problem except that they used different notations, whether it is node-identifier based or link-demand-path-identifier based.

## 2.3 The Appropriateness of NFP

This section is basically a summary of network flow problem. NFP can be used in many network design problems. Simply put, if given the required demand volume, a network provider wants to determine how much resources is needed and how to allocate them economically within particular set of flow/routing constraints. This is widely known as *uncapacitated design* [9]. In contrast, once the capacity of links are given as well as the demand volume, the problem changes to distribute flows on various different paths so that the given network goal can be optimized as much as possible and even without any objective, feasibility of a network is still what a provider needs to face. This is called *capacitated design* [9]. Capacitated problems usually occurs in short-term network design when capacity is not able to be added to the network, while uncapacitated problems are faced in long-term network planning.

Most of these problems can be formulated as a classic multicommodity flow problem which is discussed in section 2.1. As will be seen in the remaining of this thesis, it is very useful and effective to formulate multi-layer cellular network as a capacitated problem with flexible paths. In the next section, it will introduce a network design problem—fair networks, which is the primary criterion adopted in the future practical system of this thesis.

## 2.4 Fair Networks

The new internet architecture has led to a rise of interest in designing bandwidth sharing algorithms. The objective of the algorithms is to realize high bandwidth utilization rate and at the same time maintain *fairness*, such as Max-min Fairness (MMF) and Proportional Fairness (PF) we are going to introduce.

This section consists of two parts. Firstly, the notion of fair networks; secondly, max-min fairness criterion.

### 2.4.1 Notion of Fair Networks

Fair networks are networks that have elastic demand. Here, "elastic" means that every demand can consume any amount (probably within a predefined bounds) of

resources or bandwidth assigned to its paths [9]. For example, a network that has demands creating elastic traffic can adopt to any bandwidth currently assigned to it. There are a number of ways to define fairness but one intuitive explanation is that every network demand is treated fairly.

Under the condition that the capacities of links are not exhausted, fair networks problem is, in general, how and how much to assign demand volume to every demand in this context. Recall the previous three-node network example where the constraint functions only have capacity restriction but no specific value for demand volume i.e.,  $h_d$ . This means that the demand constraint needs to be met within some bounds. A common knowledge of a solution is to assign the minimum value of the bound to the demand if the capacity limits are still satisfied; if even the lower bound can not satisfy the capacity constraints, there is no feasible solution at all. Afterwards, we certainly want to make the demand pairs carry traffic more than the lower bound in order to maintain some certain level of fairness among different demands. [10][12]

A question comes up: how to assign demand volumes in a relatively fair method? Moreover, how the throughput of the whole network is affected is also of great interest. So far, the most well known method to realize fairness is *Max-Min Fairness* (MMF) [9]. Usually, the very first step of MMF is to assign "the same maximal volume to all demands [9]" To explain this point, consider a network with three nodes,  $D = 3$  demands and  $E = 2$  links in Figure 2.2 [9] and meanwhile assume that the link capacities are  $c_1 = c_2 = 1.5$ . Apparently, all demands have only one path. Based on the principle of MMF, after the first step the flow assigned to each demand is  $x_{11} = x_{21} = x_{31} = 0.75$  and in fact, there is no more steps since this is the maximal flow that can be allocated to each demand otherwise the link would be exhausted. However, if we increase  $c_2$  to 2 then after the first assignment, there is still 0.5 available on  $e_2$ . In this case, only  $d_2$  can use the extra 0.5 (because if  $d_3$  uses that, the capacity of  $e_1$  will be exceeded) increasing its total volume to 1.25. Via this simple instance, we can obtain a rough procedure of MMF allocation:

- allocate the same maximum volume to demands, assuring that the minimal assignment is maximized.
- if after the first step, there is still free capacity on some links, continue the above process.

From users' point of view, this solution seems to be perfect. However, if the objective was to optimize the total throughput of the network, we can achieve a throughput of 3 by a very unfair allocation where  $x_{11} = x_{21} = 1.5$ ,  $x_{31} = 0$ . With MMF, the throughput is 2.25. Thus, it can be observed that what is good for the network may not be necessarily good for individual users and vice versa. *Proportional Fairness* (PF) solution [9] is a compromise between throughput maximization

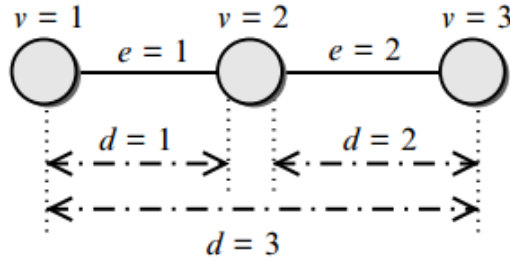


Figure 2.2: Example for illustration of MMF [9]

and MMF solution. Basically, PF changes the goal to maximize the sum of logarithms of the flows assigned to demand i.e.,  $\log x_{11} + \log x_{21} + \log x_{31}$ . With basic mathematical knowledge, it is not difficult to understand the rationale behind PF: firstly, it does not allow to assign 0 volume to demands; secondly, it makes it less beneficial to assign too much volume (think about the property of logarithm function). The final solution of proportional fairness is  $x_{11} = x_{21} = 1$ ,  $x_{31} = 0.5$  where the flow using two links (long flow)  $x_{31}$  is smaller than the flow with only one link (short flow)  $x_{11}$  and  $x_{21}$  and the total throughput with PF solution is 2.5, which is a bit larger than that of MMF.

In summary, to compare both fairness criteria: in terms of throughput, PF does better than MMF by favoring the short flow thereby leading to a less fair solution, whereas from the users' aspect, MMF is relatively fair at the expense of throughput. In order to give a deeper insight of MMF, the next subsection will briefly introduce an algorithm of MMF criterion for fixed paths.

### 2.4.2 Max-Min Fairness for Fixed Paths

In general, there are three elastic traffic (see next subsection) optimization: i) fixed paths, ii) pre-defined paths, iii) free paths. For fixed paths, a single path is defined between source and destination (O-D pair) and the task is to allocate bandwidth assigned to each demand. In pre-defined paths case, a set of available paths can be potentially used to realize the flow demand of each pair. Then the task is not only to determine the bandwidth allocation but also to figure out the specific paths that are used. In free paths case, both the bandwidth and the routes used are determined simultaneously.

In this subsection, only the solution of MMF for fixed paths will be discussed, although MMF can also effectively solve problems with capacitated flexible paths. In this situation, every demand is assigned with one single path which means  $P_d \equiv 1$  for each demand  $d$ . As has been introduced, the objective is to maximize the minimum of bandwidth allocations  $x_d$ ,  $d = 1, 2, \dots, D$  subject to capacity and satisfying non-negativity constraints. Formally, let  $\mathbf{x} = (x_1, x_2, \dots, x_D)$  be the

allocation vector sorted in non-decreasing order.  $\mathbf{x}$  is the max-min allocation as long as it is lexicographically maximal among all allocation vectors sorted in non-decreasing order. The definition of "lexicographically maximal" is that a  $n$ -vector  $\mathbf{m} = (m_1, m_2, \dots, m_n)$  sorted in non-decreasing order ( $m_1 \leq m_2 \leq \dots \leq m_n$ ) is lexicographically greater than another  $n$ -vector  $\mathbf{k} = (k_1, k_2, \dots, k_n)$  sorted in non-decreasing order if there is an index  $t$ ,  $0 \leq t \leq n$  meeting:  $m_j = k_j$ ,  $j = 1, 2, \dots, t$  and  $m_t > k_t$  [9][12]. For example, vector (3, 4, 8) is greater than (3, 3, 9) ( $t = 2$ ) and vector (4, 4, 4) is greater than (2, 20, 50) ( $t = 1$ ). In other words, the allocation is max-min optimal when there is no way to increase the allocation for some demand  $i$  at the expense of other demands with greater allocation. In addition, it can also be deduced that there exists at least one saturated link  $e$  which belongs to the route realizing a demand  $d$ , i.e.,  $\sum_d \eta_{ed} x_d = c_e$  exists.

The formulation of the above problem considered previously is as follows:

**indices**  $d \in 1, 2, \dots, D$  demands,  $e \in 1, 2, \dots, E$  links

**constant**  $\eta_{ed}$ , 1 if  $e$  belongs to the path of demand  $d$ ; 0 if not

**constant**  $c_e$ , capacity of link  $e$

**variable**  $x_d$ , flow variable of  $d$  and  $\mathbf{x} = (x_1, x_2, \dots, x_D)$

**objective** find the lexicographically maximal vector  $\mathbf{x}$

**constraint**  $\sum_d \eta_{ed} x_d \leq c_e$

**constraint**  $\mathbf{x} > 0$

According to the above analysis and background, the following algorithm can be used to find the max-min fair allocation  $\mathbf{x}$  for fixed paths problem.

### Algorithm 2.4.2

1. let  $\mathbf{x} = 0$ .
2.  $\Delta := \min [c_e / \sum_d \eta_{ed} : e \in 1, 2, \dots, E]$
3. set the following

$$c_e := c_e - \Delta (\sum_d \eta_{ed}), \quad e \in 1, 2, \dots, E$$

$$\mathbf{x} := \mathbf{x} + \Delta, \quad d \in 1, 2, \dots, D$$

do the following steps:

- Remove all links with  $c_e = 0$ .
- Remove all paths and demands that use the removed link  $e$ .

4. repeat step 1 until there is no demand remaining.

So far, the fair network problem has been well illustrated. It is worth mentioning that there is another fairness criterion called *relative fairness* (RF) [12]. It is just a special case of MMF—a weighted situation of MMF i.e., the elastic traffic is bounded (for further reading, see [12]).

Fairness, again, is of great significance in this thesis since it is the fundamental theory of the bandwidth allocation task we will solve in what follows.

### 3. FORMULATION AND SOLUTION OF HETEROGENEOUS NETWORK SYSTEMS

This chapter is the core content of the thesis and is divided into three sections. In section 3.1, the very detailed introduction to our particular problem, resource allocation among subscribers in heterogeneous network system, will be shown, including the notions and mathematical formulations. The corresponding solution to the problem is in section 3.2, where we will illustrate two algorithms, a basic one and a simplified one. The last section of this chapter gives out some details of the implementation tool—AMPL for solving the problem.

#### 3.1 NFP Formulation of Our Problem

The following content includes two parts. First, we will describe our problem more specifically and then the NFP formulation of our studied system will be illustrated.

##### 3.1.1 System Description

In the introduction part, Figure 1.1 has briefly shown what our system looks like. In order to make the system more intuitionistic, the general system of interest is shown in Figure 3.1.

##### Macro-LTE base station

As shown in Figure 3.1, a single macro-LTE basestation is located in the center of its coverage area. We assume that a single omnidirectional antenna does not affect the problem statement, although many antennas associated with a basestation are covering their segments. An aggregate rate  $C_0$  is provided by the macro station to all potential users within this circle. By assumption, neither the position of subscribers nor the propagation environment can affect the expected obtained rate. Besides, we assume that all subscribers are treated fairly by the basestation, i.e., for  $N_0$  subscribers, each will get a rate of  $C_0/N_0$ .

##### WLAN base stations

Surrounding the macro-LTE station, a number of IEEE based wireless local area networks (WLAN), say  $M_w$ , are marked as wfBS in Figure 3.1. The positions of

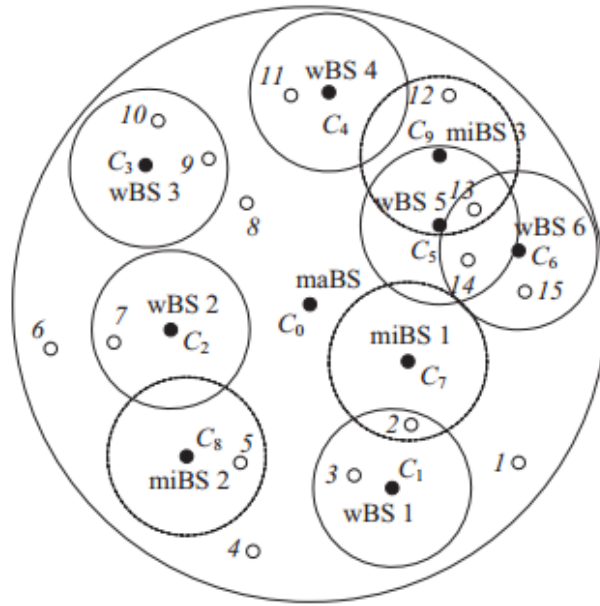


Figure 3.1: An illustration of multiaccess cellular systems

them are random but supposed to be known to the macro-LTE station and each wfBS provides a rate of  $C_i$ ,  $i = 1, 2, \dots, M_w$ . Assume that between wfBSs using the same channel there is no significant overlapping and these BSs are well planned. In fact, in the reality the above assumption is not inherently restrictive since channel selection is usually completed in a guided way or there is some built-in algorithm helping wfBS to choose the best possible channel in a certain area. Furthermore, there could be some situations where some subscribers are located on the boundary between two wfBSs, which will be addressed explicitly in the practical implementation of the system. Similarly, all wfBSs provide equal rate to the subscribers connected to them.

### Micro-LTE base stations

Besides the above two types of stations, there are  $M_l$ , randomly distributed micro-LTE stations serving some internal areas, providing a capacity unit of  $C_i$ ,  $i = 1, 2, \dots, M_l$  to all their subscribers and certainly, their positions are also clear to macro-LTE. All micro-LTE are perfectly provisioned indicating that no interference exists between them and with macro-LTE. Further, if two of them are serving the same area, they are supposed to use different frequencies.

### Subscribers, connectivity, tasks

There are, for example,  $N$  subscribers in total in the coverage of macro-LTE. They can be considered as traffic demands and all the demands are elastic with

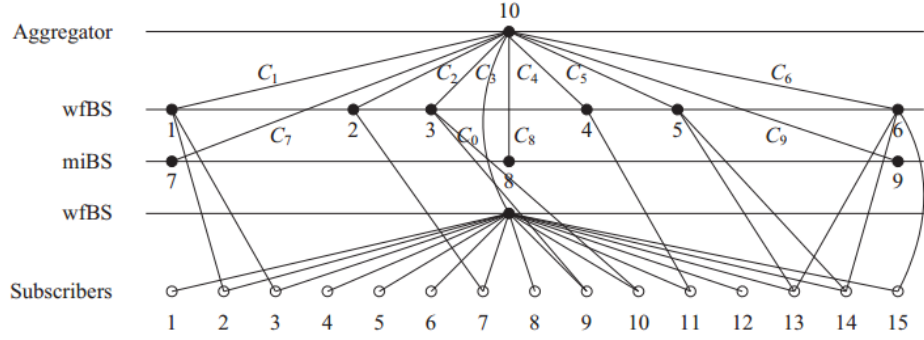


Figure 3.2: An abstract view of the system

minimum or maximum requirements of the final bandwidth or rate allocation. Recall that elastic traffic is the one occupying all the available resources provided to it. Depending on the subscribers' positions, available interfaces of terminals and locations of stations, these demands can be associated with 1) macro-LTE BS 2) none, one or more micro-LTE BSs 3) none one or more wfBSs. Generally speaking, the  $N$  subscribers is comprised of what follows:

- $N_0$  that can connect to macro-LTE stations only
- $N_{iw}$  that can be served only by WLAN stations
- $N_{il}$  that can be connected to micro-LTE stations only
- $N_{iwl}$  that can connect to both micro-LTE and wfBS
- $N_{iA}$  that can use services from all three types of BSs

Overall,  $N = N_0 + N_{iw} + N_{il} + N_{iwl} + N_{iA}$ .

Note that it is assumed that every subscriber's position is well known to the macro-LTE so that particular control information can be provided to subscribers, as to which air interface they should currently be connected to. All users are supposed to be able to connect to any wireless interface if they reside in their coverage area. Figure 3.2 is an abstract view of the same network with the one in Figure 3.1. 15 subscribers and 3 layers of access networks, in particular, 3 micro-LTE stations and 6 wifi stations.

Additionally, an aggregation node marked by 10 is also introduced. In fact, this node may exist in reality if all the equipments are provided by a single operator. In our system, it will help to define an abstracted formulation later. Some links between end nodes and base stations may not exist depending on the terminals' capability and locations of subscribers. The link rates between all the base stations

in the area and the aggregator are those of corresponding access networks. The capacity of links between end systems and BSs in this abstracted model should be at least those provided by corresponding BSs. Finally, we see that within this model there is no need for us to distinguish between micro-LTE BSs and wfBSs. The only difference between them in our context is capacity which is explicitly taken into account in our model.

So far, the system representation has been illustrated in detail. We will use this model to apply those optimization methods within the context of capacitated fair network design problem. In what follows, we will first develop a more specific topology for the system and then formulate our problem in a mathematical way.

### 3.1.2 Formulation

This subsection will be describing a 3-layer system i.e., there are 3 base station layers within the system. As the one considered previously, the base stations are not necessarily micro-LTE or WLAN station; they can be pico or femto stations. As long as the system has the above features, it can be formulated like what follows. Before continuing the formulation, a simplified version of the system will be introduced.

Below, we will again introduce what entities are included in the system:

- N subscribers
- 3 base station layers
- $N_1$  base stations at layer 1 (macro-LTE)
- $N_2$  base stations at layer 2 (micro-LTE or wifi stations)
- $N_3$  base stations at layer 3 (micro-LTE or wifi stations)
- in total,  $M = N_1 + N_2 + N_3$  BSs
- a physical aggregator

Besides, we have the following assumptions which are discussed in the last subsection.

- a subscriber may have access to one or several base stations at each layer
- traffic generated by a subscriber is greedy (full-buffer) and elastic
- the positions of subscribers are known
- rate obtained at layer 2 or layer 3 is independent on the distance to a BS
- rate of layer 1 depends on the distance to the BS

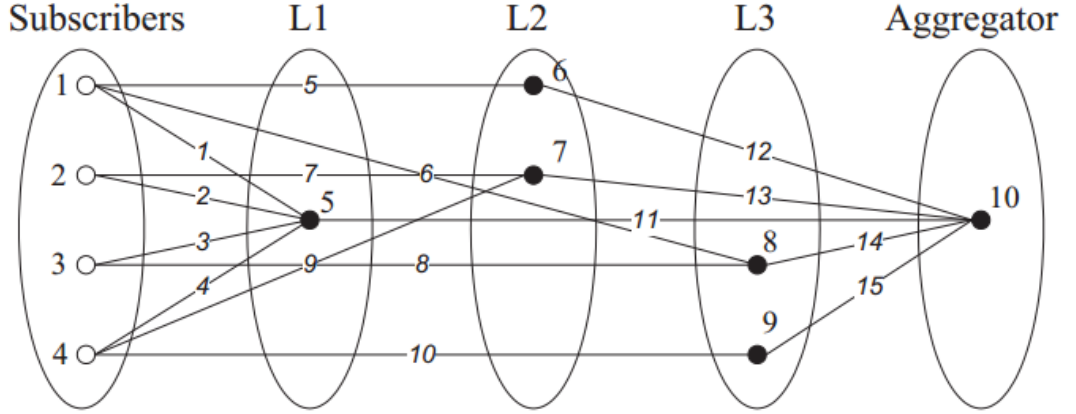


Figure 3.3: A sample topology of four-user system

- no interference affects the performance

Note that the centralized resource allocation module is integrated in layer 1 BS equipments and the decision on allocation is determined when the new users come into the system. A topology with four subscribers are shown in Figure 3.3 [4]. A user may simultaneously utilize multiple wireless interfaces, which is our concern, namely bifurcated resource allocation. It is also the reason why we need an aggregator.

By analyzing the topology above, it is concluded that defining the link rate between users and BSs is not possible since they are actually shared links. Further, the system is redundant because the capacity of the link connecting a BS and the aggregator should be equal or greater than that offered by the BS. For instance,  $c_{12} \geq c_6$ ,  $c_{13} \geq c_7 + c_9$ ,  $c_{14} \geq c_6 + c_8$ , and so on. The links like  $e_{12}$ ,  $e_{14}$  and  $e_{13}$  bring no constraint and are redundant. Therefore, if we remove them, a simplified topology can be generated.

### Simplified model

Figure 3.4 is the simplified topology in which the redundant links are removed. Node 1 is added to be a logical aggregator while node 2 is the physical aggregator, node 10 in Figure 3.3. Although the possible paths between origin (O) and destination (D) are not immediately defined, we can still clearly identify the routes by providing it with the set of possible paths between these O-D pairs. However, it is still very difficult to explicitly determine the rates of the links  $e = 1, 2, \dots, N$  to node 1 since they are basically subflows that realizes user demands.

To solve the rate representation problem, we develop the final topology of this model, shown in Figure 3.5. All demands are within the logical node 1 and physical node 2 and the number of links is the same as the number of base stations, in our case  $M = N_1 + N_2 + N_3$ , while their capacities equals to that of the corre-

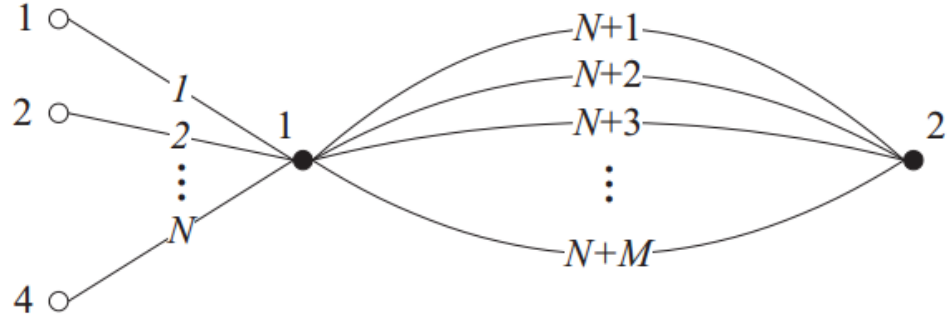


Figure 3.4: A sample topology of four-user system

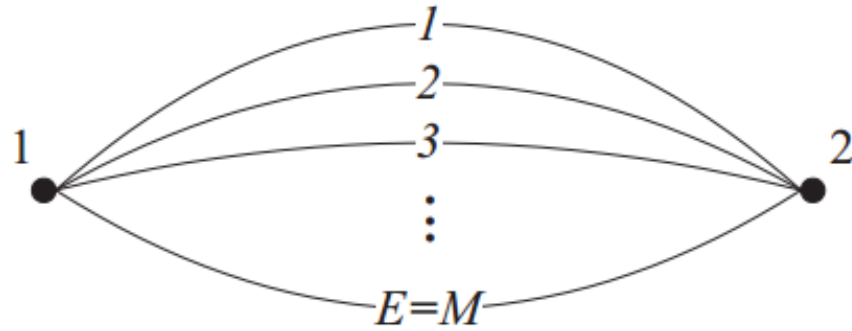


Figure 3.5: Final model of the topology

sponding base stations. Further, some of the links are shared by multiple demands but not all of them. Paths for the demands are clear to us and flows that must be realized to meet the subscribers' demands are unambiguously defined by these paths.

### Formulation

As we have seen, there are two nodes, logical and physical aggregators, in the system. What connects them are  $E = N_1 + N_2 + N_3$  links where  $N_i$  is the number of base stations at layer  $i$  and each layer has at least one user associated with. Due to the fact that the distance between subscribers and base stations has no effect on the resource provided, link rates correspond to that supplied by the base stations only. Recall that we are interested in finding a bifurcated or split solution.

The formulation starts from the  $N$  demands that need to be realized between the two aggregators. The notation method introduced in chapter 2 will be utilized in our formulation. If the demands are marked with  $d = 1, 2, \dots, N$ , then the demands volumes are represented as:

$$h_d, d \in 1, 2, \dots, N \quad (3.1)$$

and they are to be determined (unknown in advance) with the assumption that the

traffic is elastic and greedy.

Let  $\Gamma_d$  be the set of paths for demand  $d$ ,

$$\Gamma_d = \{P_{d1}, P_{d2}, \dots, P_{dp_d}\} \quad (3.2)$$

$p_d$  is the number of available paths of demand  $d$  i.e., in our case the number of base stations a demand is associated with. The subsets  $P_{dp_d}$  consist of exactly one link which connects the logical aggregator and physical aggregator in our topology. For example, if demand 4 is in the coverage of macro-LTE, wifi station and micro-LTE station, then it has three paths to realize its demand requirement, which are 1, 2, 3 representing macro-LTE, wifi station and micro-LTE respectively. Therefore, its paths set is then  $\Gamma_4 = \{P_{41}, P_{42}, P_{43}\}$  where  $P_{41} = \{1\}$ ,  $P_{42} = \{2\}$  and  $P_{43} = \{3\}$ .

The flow variables, the amount of flow assigned to demand  $d$  is denoted as

$$x_{dp}, \quad d \in \{1, 2, \dots, N\} \quad p \in \{1, 2, \dots, p_d\} \quad (3.3)$$

Thus we get the demand constraints which are

$$\sum_{p=1}^{p_d} x_{dp} = h_d, \quad d \in \{1, 2, \dots, N\} \quad (3.4)$$

where, as mentioned above,  $h_d$  is the demand volume to be obtained. This constraint indicates that the demand volume  $h_d$  of a demand  $d$  is realized via assigning flow to the available paths  $p = 1, 2, \dots, p_d$  for this demand.

Further, in subsection 2.4.2 when we introduced MMF fairness criterion, a variable  $\eta_{ed}$  was used to represent whether or not a link  $e$  belongs to a path of a demand. Similarly, we will again develop a link-path-incidence variable  $\delta_{edp}$  to help the formulation.

$$\delta_{edp} = \begin{cases} 1 & e \cap P_{dp} = e, \\ 0 & \text{otherwise.} \end{cases} \quad (3.5)$$

If a link  $e$  belongs to the path  $P_{dp}$  of demand  $d$ ,  $\delta_{edp} = 1$ . With  $\delta_{edp}$ , the capacity constraint is written as:

$$\sum_{d=1}^N \sum_{p=1}^{p_d} \delta_{edp} x_{dp} = c_e, \quad e \in \{1, 2, \dots, E\} \quad (3.6)$$

In equation (3.6),  $c_e$  is the capacity of links connecting two aggregators. In reality, it can be the link rates corresponding to the aggregate rate of respective base stations. An implication can be observed from these constraints that no link

will be overloaded by the demands which are using it in the network. As the task has been stated as *capacitated problem*, we use equalities in the capacity constraint instead of inequalities as in the previous sections.

In the section of Fair Networks we mentioned that the objective of max-min fairness criterion is to find a lexicographically maximal vector  $\vec{x}$ . The optimization task of our problem is to lexicographically maximize the demand volume variable vector  $\mathbf{h} = (h_1, h_2, \dots, h_N)$  sorted in non-decreasing order so that both (3.4) and (3.6) are satisfied and  $x_{dp}$  are continuous and non-negative. From the optimization point of view, the problem is classified as a typical linear programming (LP) problem.

In summary, the formulation for our problem is

<b>indices</b>	$d \in \{1, 2, \dots, N\}$ $p \in \{1, 2, \dots, p_d\}$ $e \in \{1, 2, \dots, E\}$
<b>constants</b>	$\delta_{edp} = 1$ if $e$ belongs to the path $p$ of demand $d$ ; 0 otherwise. $c_e$ link capacity
<b>variables</b>	$x_{dp}$ flow assigned to path $p$ of demand $d$ ; otherwise, it equals to 0 $h_d$ total bandwidth allocated to demand $d$ , $\mathbf{h} = (h_1, h_2, \dots, h_N)$
<b>objective</b>	to lexicographically maximize the demand volume variable vector $\mathbf{h} = (h_1, h_2, \dots, h_N)$ sorted in non-decreasing order
<b>constraints</b>	$\sum_{p=1}^{p_d} x_{dp} = h_d, d \in \{1, 2, \dots, N\}$ $\sum_{d=1}^N \sum_{p=1}^{p_d} \delta_{edp} x_{dp} = c_e, e \in \{1, 2, \dots, E\}$ all $x_{dp} \geq 0$

### 3.2 Solution Algorithm

Having successfully formulated the 3-layer cellular network problem, we now focus on the solution algorithm. Recall that in subsection 2.4.2 we introduced algorithm 2.4.2 for solving fixed single-path problem. However, the solution will be substantially more complicated although it is an extension of solution 2.4.2. It will contain an iteration of algorithm 2.4.2, interlaced with checking which demand allocation  $h_d$  can be further increased. As opposed to the unique result of fixed single-path problem, the problem at hand may have multiple solutions.

The extended problem (3.2.1) is as follows:

*maximize:*  $\Delta$

*subject to:*

$$\sum_{p=1}^{p_d} x_{dp} = h_d, \quad d \in \{1, 2, \dots, N\} \quad (3.7)$$

$$\Delta - h_d \leq 0 \quad (3.8)$$

$$\sum_{d=1}^N \sum_{p=1}^{p_d} \delta_{edp} x_{dp} = c_e, \quad e \in \{1, 2, \dots, E\} \quad (3.9)$$

$$\text{all } x_{dp} \geq 0, \quad d \in \{1, 2, \dots, N\}, \quad p \in \{1, 2, \dots, p_d\} \quad (3.10)$$

Initially, we need to perform "water-filling" to detect the maximal allocation that can be assigned to the demands. It can be obtained by solving problem (3.2.1). In the remaining of this section, two algorithms will be illustrated. Both algorithms are based on the well-known *non-blocking test* for checking whether there is still  $h_d$  which can be increased. The efficiencies of the algorithms depend highly on the non-blocking test (NBT). Basically, what we will conclude next is that algorithm (3.2.1) with NBT1 is less efficient and more complex than algorithm (3.2.2) but still worth introducing as it is the fundamental principle for solving capacitated flow problem, whereas algorithm (3.2.2) using NBT3 is based on dual variables of particular constraints. And meanwhile, in fact, it is also the algorithm we use in the implementation procedure since from the implementation point of view, it is more convenient and easy to be implemented (refer to pp.319 and pp.321 of book [9] for more details about the two algorithms).

In the following, we will give our detailed illustrations of two feasible algorithms and compare them meanwhile.

### Algorithm 3.2.1

- Step 1:** Let  $(\Delta^*, \mathbf{x}^*, \mathbf{h}^*)$  be the optimal solution to (3.2.1). Set  $Z_0 := \emptyset$ ,  $Z_1 := \{1, 2, \dots, N\}$ ,  $\Delta_d := \Delta^*$  for each  $d \in Z_1$ .
- Step 2:** Perform the NBT1: consider each  $d \in Z_1$  to check whether the total allocation  $h_d$  can be made bigger than  $\Delta^*$  without making any decrease to the allocation of other demands  $d'$ . If there is no blocking demand  $d$  ( $h_d$  can be further increased meaning that demand  $d$  is not blocking), then go to **Step 3**. When the first blocking demand  $d$  is detected, delete it from  $Z_1$  and put it into  $Z_0$ . If set  $Z_1$  becomes  $\emptyset$ , then stop and  $\mathbf{h} = (h_1, h_2, \dots, h_d)$  is the optimal solution or allocation vector; else go to **Step 3**. Note that  $Z_0$  is basically the set of blocking demands.

**Step 3:** Solving the following linear programming problem:

*maximize:*  $\Delta$

*subject to:*

$$\sum_{p=1}^{p_d} x_{dp} = h_d, \quad d \in \{1, 2, \dots, N\} \quad (3.11)$$

$$\Delta - h_d \leq 0, \quad d \in Z_1 \quad (3.12)$$

$$\Delta_d - h_d \leq 0, \quad d \in Z_0 \quad (3.13)$$

$$\sum_d \sum_p \delta_{edp} x_{dp} \leq c_e, \quad e \in \{1, 2, \dots, E\} \quad (3.14)$$

$$\text{all } x_{dp} \geq 0, \quad d \in \{1, 2, \dots, N\}, \quad p \in \{1, 2, \dots, p_d\} \quad (3.15)$$

**Step 4:** By solving the above LP problem, we get a new optimal solution  $(\Delta^*, \mathbf{x}^*, \mathbf{h}^*)$ . Again, put  $\Delta_d := \Delta^*$  for  $d \in Z_1$  and repeat **Step 1**.

In **Step 2** the so-called NBT1 is to solve the following LP problem for each  $d \in Z_1$ :

*maximize:*  $h_d$

*subject to:*

$$\begin{aligned} \sum_p x_{d^*p} &= h_{d^*}, \quad d^* \in \{1, 2, \dots, N\} \\ \Delta_{d^*} - h_{d^*} &\leq 0, \quad d^* \in \{1, 2, \dots, N\} \\ \sum_{d^*} \sum_p \delta_{ed^*p} x_{d^*p} &\leq c_e, \quad e \in \{1, 2, \dots, E\} \\ \text{all } x_{d^*p} &\geq 0. \end{aligned}$$

A positive outcome of NBT1 means that demand  $d$  is not blocking if the optimal solution  $h_d$  is strictly greater than  $\Delta^*$  [9]. Otherwise, the demand  $d$  considered is to be blocking and its allocation cannot be increased further. Besides, it is obvious that the calculation starts by using the results obtained from **Step 1** and **Step 3**.

However, if we are dealing with networks with a great number of nodes or demands, the non-blocking test can be rather time-consuming and inefficient. This is one reason why we choose algorithm (3.2.2) (see below) as the feasible one. Another reason is that it can be complicated to code algorithm (3.2.1) compared to algorithm (3.2.2) because there are two linear programming problems, making the practical coding work more complex than it should be.

As mentioned above, NBT3 is the most efficient algorithm based on duality of

constraints. To illustrate, let the vector  $\vec{\gamma} = (\gamma_1, \gamma_2, \dots, \gamma_N)$  be the optimal dual variables of equation (3.8) or the vector  $\vec{\gamma} = (\gamma_d : d \in Z_1)$  corresponding to constraint (3.12). According to the dual theory, if the dual variable of problem (3.2.1) satisfies  $\gamma_d > 0$  then the corresponding demand  $d$  is blocking meaning that its total bandwidth allocation  $h_d$  cannot be increased any more. Based on this property, it can be used in **Step 2** of Algorithm (3.2.1), thereby generating Algorithm (3.2.2). Hence,  $\gamma_d > 0$  implies that  $d$  is blocking, i.e.,  $d \in Z_0$  after the first execution of **Step 2**, while  $\gamma_d = 0$  does not generally indicate  $d$  is non-blocking. [4;9;12]

The modification of Algorithm (3.2.1), referred to as Algorithm (3.2.2) is:

### Algorithm 3.2.2

**Step 1:** Set  $Z_0 := \emptyset$ ,  $Z_1 := \{1, 2, \dots, N\}$ ,  $\Delta_d := 0$  for all demands

**Step 2:** Solve the LP problem below:

**maximize:**  $\Delta$

**subject to:**

$$\sum_{p=1}^{p_d} x_{dp} = h_d, \quad d \in \{1, 2, \dots, N\} \quad (3.16)$$

$$\Delta - h_d \leq 0, \quad d \in Z_1 \quad (3.17)$$

$$\Delta_d - h_d \leq 0, \quad d \in Z_0 \quad (3.18)$$

$$\sum_d \sum_p \delta_{edp} x_{dp} \leq c_e, \quad e \in \{1, 2, \dots, E\} \quad (3.19)$$

$$\text{all } x_{dp} \geq 0 \quad (3.20)$$

Let  $\Delta^*$  and  $\gamma_d$  be the optimal solution and dual variables corresponding to (3.17) respectively.

**Step 3:** Put  $\Delta_d = \Delta^*$  for each  $d \in Z_1$ . Then let  $Z_0 := Z_0 \cup \{d \in Z_1 : \gamma_d > 0\}$  and  $Z_1 := \{d \in Z_1 : \gamma_d = 0\}$ . If set  $Z_1$  becomes empty, stop and  $\mathbf{h} = (h_1, h_2, \dots, h_d) = (\Delta_1, \Delta_2, \dots, \Delta_d)$  is the final optimal solution; otherwise, go to step 2.

**Remark:** If  $\Delta^*$  obtained in step 2 is strictly larger than that from the previous iterations, then all demands in set  $Z_1$  is non-blocking; else, probably one or more  $d$  should be blocking and  $\Delta^*$  cannot be increased. Another important observation is that there will be at least one  $\gamma_d > 0$  among the newly obtained dual variables  $\gamma_d$ . The reason is because of the property of dual variables corresponding to (3.17) that  $\sum_{d \in Z_1} \gamma_d = 1$  and  $\gamma_d \geq 0$  for  $d \in Z_1$  [9].

In comparison, Algorithm (3.2.2) is much more brief than Algorithm (3.2.1) due to using dual variables. We will not prove the rationality of it in this thesis but it can be found in section 13.1.2 in [9]. The reader will observe in the next section that Algorithm (3.2.2) is much more efficient and powerful in our adopted solving tool AMPL and even for larger networks with thousands of demands, it will work perfectly quickly since it has less time consumed in non-blocking test. In addition, we only need to solve one LP problem compared to two such problems in Algorithm (3.2.1), which also dramatically reduces the coding complexity in AMPL. Furthermore, it is known that the maximum iterations needed to achieve the final solution equals to the number of BSs in our case.

### 3.3 AMPL As a Solving Tool

This section mainly introduces the tool/software we are using for solving the above linear programming problems —AMPL. AMPL is a modeling language for mathematical programming developed by Robert Fourer, David M. Gay, and Brian W. Kernighan at AT&T Bell Laboratories [11]. In order to make the readers fully understand the implementation of our model in AMPL, it is necessary to have an introduction to what this language is and how it is executed. The first subsection will fulfil this task. After introducing this background, we will show how to use AMPL to solve Max-Min fair allocation problem in the heterogeneous network with three-tier topology.

#### 3.3.1 Background of Mathematical Programming

AMPL as a modeling language was invented to help people use computers to solve mathematical programming models. The subject "mathematical programming" is to optimize a function with many variables subject to one or more constraints. As the objective behind the design of AMPL, it is necessary to briefly explain the background and development of mathematical programming problems.

##### **Mathematical programming**

The terminology programming was used to describe the schedule of activities. The programmers found that it is feasible to represent the amount of activity as a *variable* to be determined. The inherent restrictions can be mathematically described as a set of equalities or inequalities related to the variables called *constraints*. Then a solution satisfies all these constraints is considered as an appropriate plan. However, it turns out that for a complex operation, only specifying constraints is not enough to model them: if there are too few constraints, many solutions exist; if there are too

many, no solution can be found in the worst case. Thus, the success of programming also depends on an *objective* in addition to constraints. An objective function regarding to variables is basically used to decide which one is the best among all those feasible solutions. For example, the network flow example in chapter 2 may have quite a few feasible solutions but the optimal one that satisfies the objective "minimize the total routing cost" is unique. So far, *mathematical programming* mainly describes the problems with the objectives of many variables to minimize or maximize something subject to constraints on variables. [11]

A typical case of mathematical programming is: that in which the objective is a linear function and the constraints are linear equalities and inequalities. Such problems are called *linear program* and solving process is called *linear programming*. This is what we have mentioned many times in the previous chapters and sections and recall that in particular, the problem we are dealing with is a linear programming problem. The concept of linear programming is of great significance because various of mathematical problems can be modeled as a linear program and there are many fast and reliable means to solve linear programming problems and besides, its idea is also helpful to analyse and solve other mathematical programs that are non-linear. [11]

Like all other programming language, solving linear programs also needs a computer. This is why most of the researches on linear programming were developed since the late 1940's when computer science was available for scientific computing. The first and most well-known computational method was called simplex algorithm developed in 1947. Although linear programming is widely applied, the linearity assumption may not be realistic. In case that some functions of variables are non-linear, these problems are called *non-linear program*. Solving these problems is much more difficult and researchers have been studying them for over two centuries but the computational methods were only developed in recent years. Thus, to distinguish from the traditional optimization topics in mathematics field, mathematical programming is also known as "*large scale optimization* [11]". The subject of linear programming can also be broken down. When some variables must be numbers or integral values, the problem is called *integer programming*. Again, these problems become much trickier but with the more advanced tools developed and faster computers, large integer programs are getting increasingly tractable.[11]

### 3.3.2 Introduction to AMPL Development Environment and Language Basics

Traditional mathematical programming problems are not like other computer programming problems that only require to run some algorithm and print out the

solution. The basic procedures are like this:

- Formulate a model—an abstract system of variables, constraints, objective functions.
- Specify data as the instance of the problems.
- Create a specific objective function and constraint equations.
- Solve the established problem by running a solver, to apply the algorithm finding the best solution.
- Analyze the results
- Redefine the data and model and repeat if necessary

Things can be more straightforward if we could handle mathematical programming like what a solver does. However, the reality is that the way a human understand a problem is very different from the form a solver works with it. That is why a modeling language comes up. A modeling language is developed to express a modeler's form of modeling a problem in a computer recognized way so that it can be directly input to the system. After this, it is the computer's job to perform the phase of translation to the algorithm's form. Modeling languages make mathematical programming more efficient and reliable and in particular, make it easier to build new models and documentation of original models. [11]

AMPL is a type of modeling language and especially it is an *algebraic modeling language* that uses classical mathematical notation to represent objective and constraint functions, providing computer-readable notations. AMPL is famous for its similarity of arithmetic expressions and ordinary algebraic notation and expression of its sets and subscripts. It also offers an interactive environment. A user can switch between various solvers so the performance can be improved and once the optimal solution is found, people can view and analyze it in a modeler's way. [11]

Figure 3.6 shows a General environment interface of AMPL software. As we can see, on the right of the interface, users can edit AMPL files like .mod, .dat, .run and so on. In the middle of it, it is the command area where a user input corresponding commands so that the software can load model and data files to its system and after receiving the command `solve`, it starts to apply the internal algorithm for this linear program, MINOS solver in this figure. Since this modeling language and software may not be familiar to some readers and in order to make our formulated problem understandable, in what follows we have an concise explanation to the critical components of AMPL.

### **A linear programming example model in algebraic form**

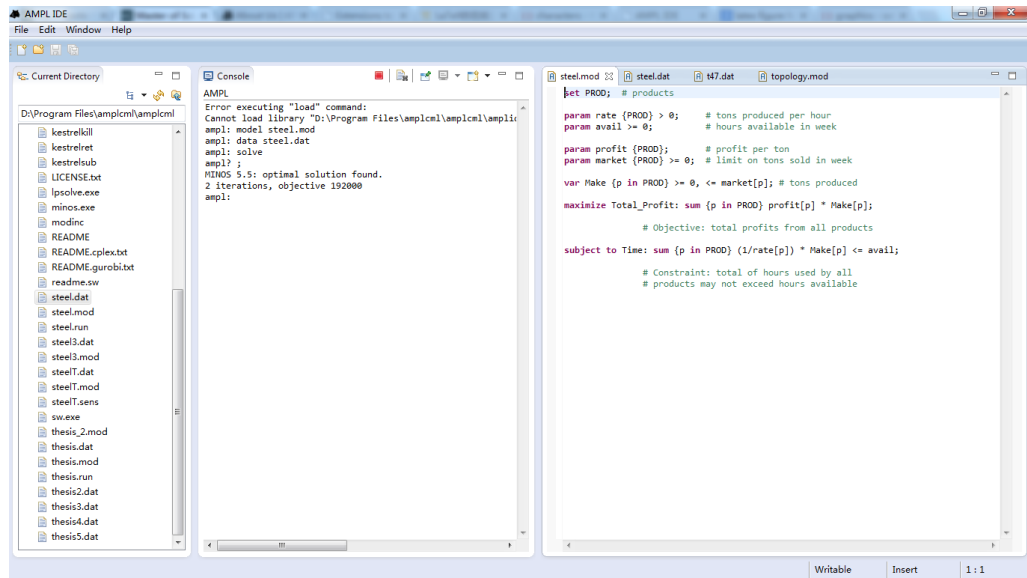


Figure 3.6: General environment interface of AMPL software

Let's take into account a product problem [11]. Since the problem has a few products and a few constraints, it is hard to illustrate with pictures and if it changes frequently, it is also hard to update. So first we need to use a compact description for the particular problem, which we call a *model* with algebraic notations for objectives and constraints. A typical linear programming model is like this:

- What is given:**
- $P$ , a set of products
  - $a_j$ , how many tons per hour of product  $j$  ( $j \in P$ )
  - $b$ , hours available at the production line
  - $c_j$ , profit of product  $j$  per ton ( $j \in P$ )
  - $u_j$ , maximal tons of product  $j$  ( $j \in P$ )
- Variables:**
- $X_j$ , amount of product  $j$  to be produced
- Objective:**
- Maximize the profit:  $\sum_{j \in P} c_j X_j$
- Constraints:**
- $\sum_{j \in P} \frac{X_j}{a_j} \leq b$
  - $0 \leq X_j \leq d_j$ , for every  $j \in P$

The above model has described thousands number of the related optimization problem. If the model are provided with specific data, it becomes a individual problem; each different collection of data will make a different instance of model. With the ability to describe long linear programs with a short model, it achieves the compromise between brevity and comprehension and is easy to be converted to a language computers are able to process. In fact, the most difficult task in a real

situation is to model correctly and provide accurate data; for solving a problem, it only depends on the solver and computing power.

### A linear programming example model in AMPL

Intentionally, AMPL is a language that is as close to the algebraic form as it can get while being easy to input through ordinary keyboard and be processed by a computer. AMPL provides constructions for all the elements listed above: sets, parameters, variables, constraints, objectives and methods to write arithmetic expressions: sums over a set and so forth. Now we will give out how the product problem is represented in AMPL.

The keyword `set` declares a set: `set P;`. The member of  $P$  is provided in the data file (see below).

Keyword `param` defines parameters like: `param b;`. It can also define a collection of values indexed by a set. For example,  $a_j, j \in P$  is expressed as `param a {j in P};`. Similarly,  $c_j, j \in P$  is `param c {j in P}` and  $u_j, j \in P$  is `param u {j in P}` respectively. AMPL uses square brackets to write subscripts in algebraic notation:  $a_j \longleftrightarrow a[j]$ .

`var X {j in P}` names the variable  $X_j$  whose value is to be determined by the solver.

The objective function is expressed as this:

```
maximize Total_Profit:  sum {j in P} c[j]* X[j];
```

`Total_Profit` is only a name of the objective function and any name will work.

Lastly, the constraints are given by

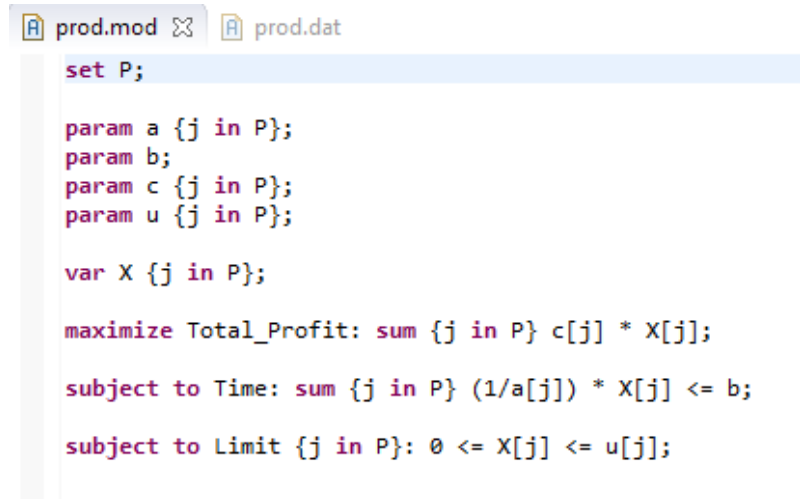
```
subject to Time:  sum {j in P} (1/a[j])* X[j] <= b;
```

```
subject to Limit {j in P}:  0 <= X[j] <= u[j];
```

Note here, `{j in P}` is *indexing expression*. This expression is of great importance since it can not only be used in declaring parameters or variables but also in any context where the algebraic model has something like *any*  $j \in P$ .

Additionally, the layout of AMPL is very free. Sets, variables and parameters must be declared before they are in use, which is like many programming language. A statement must be ended with semicolons and can be split for readability. Traditional mathematical notations have been adapted in AMPL. For instance, AMPL uses `in` rather than  $\in$  for membership of a set and `sum` instead of  $\sum$  for summation. Certainly, AMPL has a very precise grammar like many other computer languages and there are many rules and skills when defining the components in models and data but we will not explain the grammar too much. When we introduce the AMPL model of our problem, some necessary grammar details will be illustrated.

Figure 3.7 and 3.8 are the model and data files of the above product problem. Figure 3.9 shows the command window where the software loads `prod.mod` and



```

prod.mod prod.dat
set P;

param a {j in P};
param b;
param c {j in P};
param u {j in P};

var X {j in P};

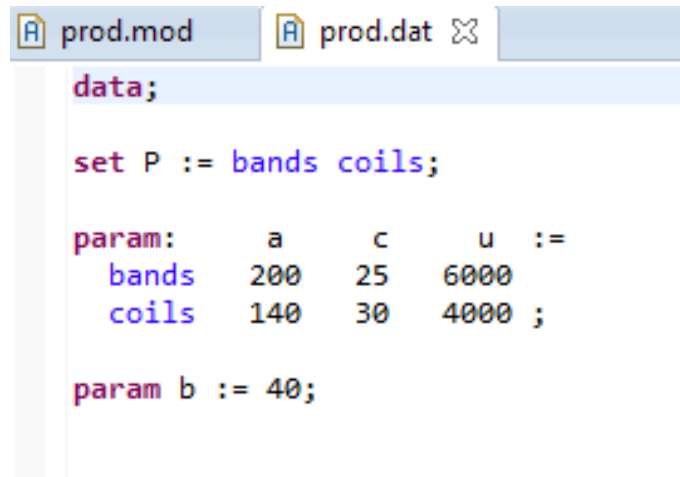
maximize Total_Profit: sum {j in P} c[j] * X[j];

subject to Time: sum {j in P} (1/a[j]) * X[j] <= b;

subject to Limit {j in P}: 0 <= X[j] <= u[j];

```

Figure 3.7: Model file of product problem



```

prod.mod prod.dat
data;

set P := bands coils;

param:      a      c      u      :=
bands      200    25     6000
coils      140    30     4000 ;

param b := 40;

```

Figure 3.8: Data file of product problem

prod.dat and solves the problem. It is worth to mention that a feature of AMPL is that it can display any set, parameter and variable with command `display`.

This example is very helpful to understand the meanings of the elements of our AMPL model. As mentioned above, in our case there will be more advanced uses of AMPL language such as loops applied in commands so without this example, it would be more difficult to catch.

### 3.3.3 AMPL Application in Solving 3-tier Heterogeneous Network Problem

With the basic knowledge of AMPL language, we can now talk about how to use AMPL to solve our concentrated network problem. In the beginning of this chapter, we illustrated the mathematical formulation of our network resource allocation prob-

```

Console
AMPL
Error executing "load" command:
Cannot load library "D:\Program Files\amplcml\amplcml\ampli
ampl: model prod.mod;
ampl: data prod.dat;
ampl: solve;
MINOS 5.5: optimal solution found.
2 iterations, objective 192000
ampl: display a;
a [*] :=
bands 200
coils 140
;

ampl: display c;
c [*] :=
bands 25
coils 30
;

```

Figure 3.9: Command window

lem and together with the formulation, the solution algorithms were also introduced in detail. Recall that we mentioned that the algorithm we adopted is Algorithm (3.2.2) because of its convenience. First of all, the algebraic form will be elaborated.

- **Building model file**

We first focus on the LP problem in Algorithm (3.2.2):

**maximize:**  $\Delta$

**subject to:**

$$\sum_{p=1}^{p_d} x_{dp} = h_d, \quad d \in \{1, 2, \dots, N\}$$

$$\Delta - h_d \leq 0, \quad d \in Z_1$$

$$\Delta_d - h_d \leq 0, \quad d \in Z_0$$

$$\sum_d \sum_p \delta_{edp} x_{dp} \leq c_e, \quad e \in \{1, 2, \dots, E\}$$

$$\text{all } x_{dp} \geq 0$$

Combined with the Algorithm (3.2.2), what needs to be done in an algebraic form of this linear program is to list the elements required.

**Given:**  $Z_1$ , the collection for non-blocking demands

$Z_0$ , the set for blocking demands

(Both of the two sets are dynamic during the process of the algorithm)

$N$ , the number of subscribers

$E$ , the number of links connecting two aggregators or the number of base stations

$c_e$ , for each  $e \in \{1, 2, \dots, E\}$ , link capacity

$\delta_{edp} = 1$ , if  $e$  belongs to the path  $p$  of demand  $d$ ; 0 otherwise.

**Variables:**

$x_{dp}$ , flow variables assigned to demand  $d$  on path  $p$

$h_d$ , total bandwidth allocation of demand  $d$

$\Delta$ ,  $d \in \{1, 2, \dots, N\}$ ,  $p \in \{1, 2, \dots, p_d\}$

**Other parameters:**

$\gamma_d$ , dual variable constraint (3.17)

$\Delta_d$ , equals to 0 at first and  $\Delta^*$  after iterations

**Objective:**

maximize  $\Delta$

**Constraints:**

$$\sum_{p=1}^{p_d} x_{dp} = h_d, \quad d \in \{1, 2, \dots, N\}$$

$$\Delta - h_d \leq 0, \quad d \in Z_1$$

$$\Delta_d - h_d \leq 0, \quad d \in Z_0$$

$$\sum_d \sum_p \delta_{edp} x_{dp} \leq c_e, \quad e \in \{1, 2, \dots, E\}$$

After appropriate translation to AMPL language like what was done in the previous product problem, we obtain the following form of our problem in AMPL software (see figure 3.10).

Still, the unique readability feature of AMPL makes the layout assemble to what a mathematical looks like. The basic declarations methods used in the product linear program also appears in this case. We declared three sets:  $Z_0$ ,  $Z_1$ , and  $Z_2$  among which  $Z_2$  has no actual meaning but is just a intermediate set for assignment. The collection of demands  $\{1, 2, \dots, N\}$  and collection of links  $\{1, 2, \dots, E\}$  are denoted as `set demand_nos := 1..D` and `set link_nos := 1..E`. Besides, because the limitation of keyboard, we use `r` instead of  $\Delta$  and `t_d` to replace  $\Delta_d$ . We will illustrate this later in this section.

Parameters like  $D$ ,  $E$  are defined a bit differently with the limitation to the values ( $>0$ ) and type (integer) of them. The point we did not introduce earlier is a different indexing expression. For example, `param t{demand_nos}` means that parameter `t` has subscripts indexed from set `demand_nos`. This works similarly to `link_capacity` which is  $c_e$  in mathematical mode.

```

param D > 0 integer;
param E > 0 integer;

set link_nos := 1..E;
set demand_nos := 1..D;
set Z1;
set Z2;
set Z0;
set path_nos{d in demand_nos};

param t{demand_nos};
param link_capacity {link_nos} >=0 integer;
set Path {d in demand_nos,p in path_nos[d]} within link_nos;

param delta {e in link_nos,d in demand_nos,p in path_nos[d]}
  = if e in Path[d,p] then 1 else 0;

var r >= 0;
var h {demand_nos}>=0;
var x {d in demand_nos, p in path_nos[d]} >= 0;

maximize opti: r;

subject to first_con {d in demand_nos}:
  sum {p in path_nos[d]} x[d,p] = h[d];

subject to second_con {d in Z1}:
  r - h[d] <=0;

subject to third_con {d in Z0}:
  t[d]-h[d]<=0;

subject to fourth_con {e in link_nos}:
  sum{d in demand_nos} (sum {p in path_nos[d]} (delta[e,d,p]*x[d,p]))
  - link_capacity[e]<=0;

```

Figure 3.10: Model file of the 3 -layer network problem

Two very important elements are `path_nos` and `Path`. The declaration of these two sets are collections of sets. To illustrate, `path_nos {d in demand_nos}` means that for each member  $d$  of `demand_nos` there is to be a set `path_nos[d]`. In practice, it indicates the total number of paths ( $p_d$ ) of each demand  $d$ . Because each demand has a different number of candidate paths, we have to use this technique to distinguish. Similarly, for each demand  $d$  of `demand_nos` and  $p$  of `path_nos[d]` there is a set `Path[d,p]`, a subset of `link_nos`. The set is critical since the value of parameter `delta` depends highly on it. As you can see that when declaring `delta` ( $\delta_{edp}$ ), the much same principle is used too: `param delta {e in link_nos,d in demand_nos,p in path_nos[d]}`.

There is no much to emphasize about the variable declarations if the above methods are fully understood.  $\Delta$ ,  $h_d$  and  $x_{dp}$  are defined as usual except that  $\Delta$  is now  $r$ .

From this relatively complicated model, it is obvious that AMPL has many remarkable advantages. Firstly, its readability. People can easily understand what is the meaning of each expression as long as one has some basic knowledge of algebra or calculus. Secondly, AMPL provides very convenient and flexible indexing technique even for indexing over other sets. Further, it also has some features of other programming languages like condition statements making it possible to declare constants that have multiple values.

- **Data File**

After the most important work—establishing model has been completed, we need a file to specify data for the model so that a particular instance of the problem can be defined. The statement of specifying data is different from the way to declare model elements and sometimes it can be time-consuming and exhausting. One should also be very careful while making data files especially for larger models because even though a small mistake may lead to inappropriate results or even crash a program. In fact, except for variables, all other sets and parameters need to be specified with data.

For our model, we must assign data to the following components: three parameters  $D$ ,  $E$  and  $t$ ; sets  $Z0, Z1, Z2, link\_capacity, path\_nos[d], Path[d,p]$ . The most complicated part is to specify the data for  $path\_nos[d]$  and  $Path[d,p]$ . For both of them, we need to know how many available paths there is for each demand  $d$  and for every path of  $d$ , which link is used and which is not. For simplification but not losing generalization, we choose a model instance with 20 subscribers and 9 links i.e., 9 base stations including 1 macro-LTE station, 4 wifi stations and 4 micro-LTE stations. The distribution of subscribers is random. Here to explain how to specify data, we randomly select one data file.

Figure 3.11 shows how to assign values to simple parameters and sets. Through this,  $D$  and  $E$  are assigned with value 20 and 9 respectively. Sets  $Z0$  and  $Z2$  are initially empty sets while set  $Z1$  originally contains the whole demands whose number is 20.

```
data;
param D := 20;
param E := 9;

set Z0:=;
set Z2:=;
set Z1 := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20;
```

Figure 3.11: Sample Data Statement

In Figure 3.12, a more complicated data specification method is shown. As we know,  $link\_capacity$  and  $t$  are both parameters indexed over sets. Thus, when

assigning values to them, we need to make it clear that which value is for which subscript. For the link capacity specification, the format is like that in the figure: the left column points out the links and right column is the capacities of each link. Parameter  $\tau[d]$  that is  $\Delta_d$  actually is also specified as above and its values are originally 0.

```

param: link_capacity :=
1      100
2      400
3      400
4      400
5      400
6      100
7      100
8      100
9      100 ;

param: t :=
1      0
2      0
3      0
4      0
5      0
6      0
7      0
8      0
9      0
10     0
11     0
12     0
13     0
14     0
15     0
16     0
17     0
18     0
19     0
20     0;

```

Figure 3.12: Sample Data Statement

Figure 3.13 illustrates the specification of  $\text{path\_nos}[d]$  and  $\text{Path}[d,p]$ . It is noticeable that for demands like 1,2,...,11, they have only 1 available path for realizing their demands whereas the remaining 9 demands have 2 paths for each. Then for specifying  $\text{Path}[d,p]$ , we must assign the corresponding link for each path of  $d$ . For example, demand 13 has two paths denoted as `set path_nos[13]:=1 2;` and for each of its path, link 1 and 2 are used. One may be curious about that why there are some links missing in this example (only links 1, 2, 4, 5, 6, 7 appeared). This is because the subscribers are not located in their coverage. We will explain this shortly.

- **Run file**

It may not be proper to use name "Run file" for this part. In fact, this part is going to discuss the script file containing a sequence of commands. It also allows to use programming language constructs like `for`, `if` and `repeat` to perform conditionally and repeat statements. In effect, through this AMPL let users to write small

```

set path_nos[1]:=1; set path_nos[2]:=1; set path_nos[3]:=1;

set path_nos[4]:=1;set path_nos[5]:=1;set path_nos[6]:=1;

set path_nos[7]:=1;set path_nos[8]:=1;set path_nos[9]:=1;

set path_nos[10]:=1; set path_nos[11]:=1;

set path_nos[12]:=1 2; set path_nos[13]:=1 2;set path_nos[14]:=1 2;

set path_nos[15]:=1 2; set path_nos[16]:=1 2;set path_nos[17]:=1 2;

set path_nos[18]:=1 2; set path_nos[19]:=1 2;set path_nos[20]:=1 2;

set Path[1,1] := 1;
set Path[2,1] := 1;
set Path[3,1] := 1;
set Path[4,1] := 1;
set Path[5,1] := 1;
set Path[6,1] := 1;
set Path[7,1] := 1;
set Path[8,1] := 1;
set Path[9,1] := 1;
set Path[10,1] := 1;
set Path[11,1] := 1;

set Path[12,1] := 1; set Path[12,2]:=2;
set Path[13,1] := 1;set Path[13,2] := 2;
set Path[14,1] := 1;set Path[14,2] := 2;
set Path[15,1] := 1;set Path[15,2] := 5;
set Path[16,1] := 1;set Path[16,2] := 6;

set Path[17,1] := 1;set Path[17,2] := 7;

set Path[18,1] := 1;set Path[18,2] := 4;
set Path[19,1] := 1;set Path[19,2] := 4;
set Path[20,1] := 1;set Path[20,2]:=4;

```

Figure 3.13: Sample Data Statement

programs in AMPL language. A `for` statement, for example, can make commands executed once for each member of a set.

The format of this file in our case is `.run` (see Figure 3.14), which is why we name this part as "Run file". The file "topology.run" in our situation is significant since it effectively implements the important procedure for solving our problem—the Algorithm (3.2.2). Recall that in the algorithm, there is a need to iterate and to execute over a set. That is why we need make small program in AMPL command language.

To run a script file, in the command window of AMPL software input `include filename`. After this command, the AMPL software will execute the commands in this file line by line. The beginning two commands are to load the model file "topology.mod" and one data file "t24.dat" so that our network problem model and its corresponding instance can be established. The format and content of data files

```

topology.mod topology.run t24.dat
model topology.mod;
data t24.dat;
option solver cplex;

repeat {
solve;
let {d in Z1} t[d]:=r;
for {d in Z1}
{ if second_con.dual[d] =0 then let Z2:=Z2 union {d};
}

for {d in Z1}
{
if second_con.dual[d] >0 then let Z0:=Z0 union {d};
}

let Z1:=Z2;
let Z2:={};
} until card(Z1)=0;

if card(Z1)=0
then { printf "The optimal solution is below: \n";
let {d in demand_nos} h[d]:=t[d];}

display h;

```

Figure 3.14: Command Script File

are quite similar except that several parameters or sets have different values.

Command `option solver cplex` is to change the default solver MINOS to `cplex`. CPLEX is so far the best known and most broadly used large-scale solver. Compared with MINOS, CPLEX is more efficient and more accurate in solving linear and convex quadratic programs. In our experiment procedure, we tried MINOS first but it made some mistakes in the final results because of the convexity of our problem. Then we turned to CPLEX which successfully fulfilled this task.

The loop `repeat {...} until card(Z1)=0` implements the vital part of Algorithm (3.2.2). Firstly, the problem is solved by command `solve`. By solving our linear problem, we obtain an optimal solution  $\Delta^*$ , represented as `r` and assign this value to each `t[d]` for each  $d \in Z1$ . This is done by `let d in Z1 t[d]:=r` where `let` is an assignment command.

Then we need to separate blocking demands from non-blocking demands according to the dual variables of `second_con` constraint in the model file. Two `for` statements implement this. It allows the statement within the brackets to be executed for each  $d$  in set `Z1`. The acquirement of dual variable is through `second_con.dual[d]`. The operator `union` creates a new set which is a union of two sets. Regarding the two other commands `let Z1:=Z2` and `let Z2:={}`, as we mentioned, set `Z2` is only a transition set: it just momentarily contains the non-blocking demands then be-

comes empty and reused again. Two `for` statements and the set assignment finally divide the two kind of demands.

The program checks whether the set `Z1` is empty or not by `until card(Z1)=0`. The function `card` computes the number of members in a set so `card(Z1)=0` means that set `Z1` is empty. Based on the algorithm, if `card(Z1)≠0`, the program is solved again to get a new optimal solution of  $\Delta$  or  $\mathbf{r}$  and repeat the above process to divide more blocking demands with non-blocking demands; else, stop. When set `Z1` is empty, there is no more non-blocking demands i.e., the total demands allocation can not be increased any more and the final solution is found. Lastly, we display the final allocation `h`.

Figure 3.15 shows the optimal solution, the total demand volume allocation for all demands, using a particular data file. After four steps iteration, this problem instance is perfectly solved. At first glance, the allocation seems to be unordered but if it is ordered decreasingly, we see that it is lexicographically maximal.

```

ampl: include topology.run
CPLEX 12.6.0.0: optimal solution; objective 6.66666667
4 dual simplex iterations (2 in phase I)
CPLEX 12.6.0.0: optimal solution; objective 100
25 dual simplex iterations (6 in phase I)
CPLEX 12.6.0.0: optimal solution; objective 166.6666667
5 dual simplex iterations (3 in phase I)
CPLEX 12.6.0.0: optimal solution; objective 400
4 dual simplex iterations (1 in phase I)
The optimal solution is below:
h [*] :=
1  6.66667
2  6.66667
3  6.66667
4  6.66667
5  6.66667
6  6.66667
7  6.66667
8  6.66667
9  6.66667
10 6.66667
11 6.66667
12 6.66667
13 6.66667
14 6.66667
15 6.66667
16 166.667
17 166.667
18 400
19 100
20 166.667
;
ampl:

```

Figure 3.15: Example of an optimal allocation

- **Summary**

We have spent many pages to illustrate the AMPL implementation aspect of the thesis. At first, through a simple example we introduced how AMPL modeling

language works: how the environment looks like, how to declare sets, variables and parameters. Again, this was necessary because without this illustration, one can hardly understand our model and follow how our problem is solved. In the next, we described our general model and gave the sample data file of one particular case. The implementation of the algorithm in the script file was also explained specifically. In general, AMPL can solve the 3-layer network resource allocation problem in less than 10 seconds, which is very efficient. Besides, it can be relatively easy to use for anyone who is familiar with algebraic knowledge. This is the reason why we chose AMPL as a solving tool instead of other tools like MATLAB, although they are also very fast to solve linear programming problem.

In the next chapter, we will show some our numerical results together with the related realistic topologies.

## 4. NUMERICAL RESULTS OF HETEROGENEOUS NETWORK PROBLEM

Having introduced the theoretical and implementation background, we will focus on the numerical achievements of our problem. This chapter mainly contains the topologies we established with MATLAB to approach the real cases as much as possible, the results we obtained using the former algorithm and the solving tool and finally a comparison with other two conventional heuristic schemes. In detail, 50 topologies have been produced to ensure the generalizability. For each topology, there are 3 cellular network layers, 9 base stations and 20 subscribers. We have shown the result of one particular case at the end of chapter 3 and later we will illustrate more typical ones but not all of them. At the initial stage, we encountered some problem with AMPL and got some incorrect but reasonable results. So, at first, we will explain a little about this.

### 4.1 A Simple Situation

In order to verify whether the algorithm and AMPL works correctly, we established a relatively simple but also complex enough for this objective. In general, this simple topology (see figure 4.1) include 3 base stations and 7 subscribers. The base station with largest coverage is macro-LTE and two others are wifi station and micro-LTE station respectively. All the subscribers have access to macro-LTE except that user 1 and 2 only have this way to access radio resources. Subscriber 5 is the only one that can access to all the 3 RATs i.e., it has three paths to realize its demand. For the remaining four users, each two of them are served by micro-LTE and wifi station respectively.

The model is exactly like what we showed in chapter 3 but the data will certainly be different. Figure 4.2 is the specific data of this topology. In this case, we set the capacities of stations as 100,400,300 respectively for macro-LTE ,micro-LTE and wifi station and they are referred to as 1,2 and 3 (not shown in the picture). Note that node 5 has three paths since it is within the coverage of three base stations. This data file is almost the same with the one we provided in the last chapter except for less subscribers and base stations.

As we said, this topology is so simple that we can compute it by hand and intuitively, there are several feasible solutions for this problem. However, we are

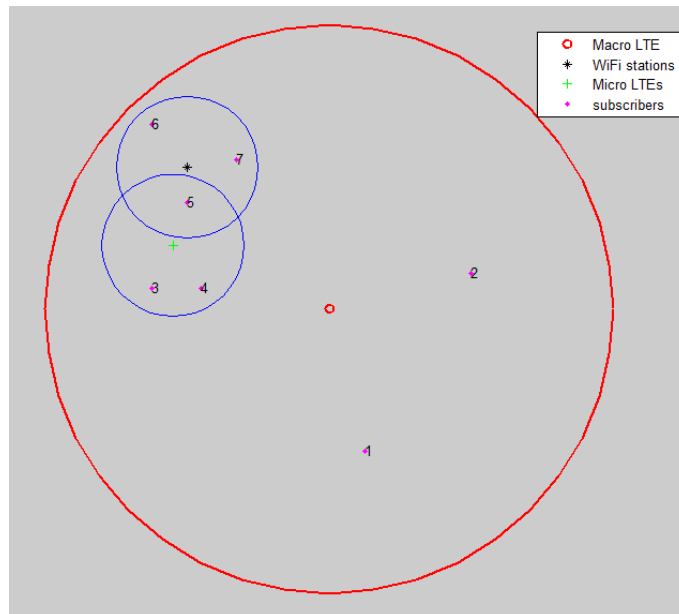


Figure 4.1: Our test topology

searching for the optimal one i.e., the one with lexicographically maximal total allocation. Figure 4.3 and 4.4 show two different results of two solvers. The left result is obtained by solver MINOS version 5.5. In reality, this may work and the link capacities are not exceeded. However, it is not the optimal one since if we compare it with the right result, it can be noticed that the allocation in figure 4 is lexicographically maximal when sorted in non-decreasing order. The right one is solved by CPLEX without exhausting any link either. Obviously, when solving the problem, MINOS did not use dual variables, which we think is the main problem that it cannot get the optimal result because the algorithm we used relies heavily on dual variables to implement non-blocking test. If we make the topology more complex, say if we add two more subscribers in the coverage of wifi station, one in the overlapping area of wifi and micro-LTE stations and one served only by macro-LTE, in total 10 subscribers, we still get the expected results where vector  $\mathbf{h} = (33.3333, 33.3333, 33.3333, 100, 100, 100, 100, 100, 100, 100)$  sorted in non-decreasing order.

So far, it has turned out that the algorithm works perfectly in realistic cases and AMPL is also appropriate for solving such problems. However, this is far from enough since we have quite a few wifi stations and micro-LTEs within a larger base station like macro-LTE and in addition, hundreds of subscribers may appear at the same time. Apparently, to show all of them in a figure is impossible and time-consuming and it is also problem to specify data in AMPL. Therefore, in the next section, we try to make our case as real as possible.

```

data;
# data for a new topology
# 1 LET station and 1 wifi station and 1 Micro LTE
# 7 subscribers
param D := 7;
param E := 3;
param: link_capacity :=
  1      100
  2      400
  3      300 ;

set Z0:= ; set Z2:=;set Z1 := 1 2 3 4 5 6 7;

param: t :=
  1      0
  2      0
  3      0
  4      0
  5      0
  6      0
  7      0 ;

set path_nos[1]:=1;
set path_nos[2]:=1;
set path_nos[3]:=1 2;
set path_nos[4]:=1 2;
set path_nos[5]:=1 2 3;
set path_nos[6]:=1 2;
set path_nos[7]:=1 2;

set Path[1,1] := 1;
set Path[2,1] := 1;
set Path[3,1] := 1; set Path[3,2]:=3;
set Path[4,1] := 1;set Path[4,2] := 3;
set Path[5,1] := 1;set Path[5,2]:=3; set Path[5,3]:=2;
set Path[6,1] := 1; set Path[6,2]:=2;
set Path[7,1] := 1; set Path[7,2]:=2;

```

Figure 4.2: Data file of our test topology

## 4.2 Numerical Results of More Realistic Cases

In this section, we will introduce more situations a network provider may face in reality. First of all, some basic parameters will be illustrated. In generally, we increased the number of base stations to 9 including: 1 macro-LTE station, 4 wifi stations and 4 micro-LTE stations. Macro-LTE has the largest coverage area which is 500 meters while wifi and micro-LTE stations both have a 100-meter coverage. Regarding to the capacities, we reset that of micro-LTE to 100 Mbps, the same with macro-LTE whereas wifi stations have much higher capacities, 400 Mbps for each. The positions of them are another problem we need to figure out. At the beginning, we assumed that macro-LTE is aware of all the positions of other BSs and both wifi and micro-LTE should be within macro-LTE coverage area. In order to make the situation more complex, each wifi and each micro-LTE are overlapping i.e, they have common areas. Additionally, there are 20 subscribers generated randomly in the coverage of macro-LTE. In terms of the topology generation, we used MATLAB as the building tool. The code for this can be found in the following.

```

ampl: include alg8.4.run;
MINOS 5.5: optimal solution found.
12 iterations, objective 50
MINOS 5.5: optimal solution found.
4 iterations, objective 150
MINOS 5.5: optimal solution found.
1 iterations, objective 175
The optimal solution is below:
h [*] :=
1  50
2  50
3  150
4  150
5  50
6  175
7  175
;

```

```

ampl: include alg8.4.run;
CPLEX 12.6.0.0: optimal solution; objective 50
5 dual simplex iterations (1 in phase I)
CPLEX 12.6.0.0: optimal solution; objective 140
1 dual simplex iterations (0 in phase I)
The optimal solution is below:
h [*] :=
1  50
2  50
3  140
4  140
5  140
6  140
7  140
;

```

Figure 4.3: Solution with MINOS solver Figure 4.4: Solution with CPLEX solver

```

+++++
% MATLAB code for generating the topologies

clear all;
clc;

sita=0:pi/20:2*pi;
% radius of BSs:
r1=500; r2=100; r3=100;

% coordinator of macro-LTE:
x0 = 400; y0 = 500;
% coordinator of wifis in total 4:
x1 = 200; y1 = 700; x2=180; y2=350;
x3=600; y3=680; x4=590; y4=300;
% coordinator of micro-LTEs in total 4:
t1=180; m1=590; t2=260; m2=300; t3=500;m3=800;
t4=690; m4=420;

% uniformed distributed points in a circle for macro LTE:
scr_x3_macrolte=unifrnd(x0-r1,x0+r1,20,1);
c=sqrt(r1^2-abs(scr_x3_macrolte-x0).^2);
scr_y3_macrolte=unifrnd(-c+y0,c+y0,20,1);

% plot the macro-LTE station marked with red empty circle
h1 = plot(x0,y0,'ro',x0+r1*cos(sita),y0+r1*sin(sita),
          'r','LineWidth',2);

```

```
text(x0,y0,'1');
hold on;

% plot four wifi stations marked with black star
h2 = plot(x1,y1,'k*',x1+r2*cos(sita),y1+r2*sin(sita));
text(x1,y1,'2');
hold on;

plot(x2,y2,'k*',x2+r2*cos(sita),y2+r2*sin(sita));
text(x2,y2,'3');
hold on;

plot(x3,y3,'k*',x3+r2*cos(sita),y3+r2*sin(sita));
text(x3,y3,'5');
hold on;

plot(x4,y4,'k*',x4+r2*cos(sita),y4+r2*sin(sita));
text(x4,y4,'4');
hold on;

% plot four micro-LTE stations marked with green '+'
h3 = plot(t1,m1,'g+',t1+r3*cos(sita),m1+r3*sin(sita));
text(t1,m1,'6');
hold on;

plot(t2,m2,'g+',t2+r3*cos(sita),m2+r3*sin(sita));
text(t2,m2,'7');
hold on;

plot(t3,m3,'g+',t3+r3*cos(sita),m3+r3*sin(sita));
text(t3,m3,'9');
hold on;

plot(t4,m4,'g+',t4+r3*cos(sita),m4+r3*sin(sita));
text(t4,m4,'8');
hold on;

% plot the 20 randomly created subscribers marked with pink dot
h4=plot(scr_x3_macrolte,scr_y3_macrolte,'m.');
```

```

legend ([h1(1),h2(1),h3(1),h4(1)],
        'Macro-LTE', 'WiFi stations', 'Micro-LTEs', 'subscribers');
axis equal;
hold off;
axis off;

```

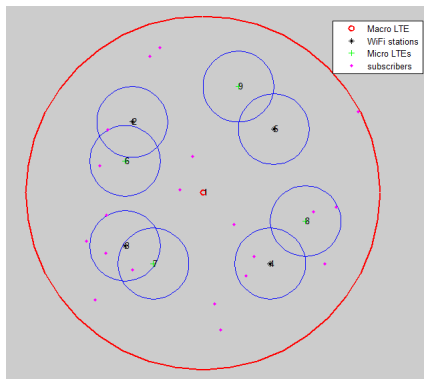
```

+++++

```

Note that we have set the coverage of each station but as we assumed earlier, both the uplink and downlink transmission are independent with the distance i.e., no matter how far a subscriber is from its corresponding BS, the possibility for receiving the signal and signal power it receives are the same with other ones. However, since we only have 20 subscribers in this large space, when we generated them randomly, there were just a few of them located at the area of WiFi or micro-LTE and in some cases none of them was there. So basically this is the reason why we are going to choose the most typical topologies where there are at least reasonable number of subscribers located in WiFi and micro-LTEs for us to analyze.

In the rest of this chapter, we will show some topology figures (starting from figure 4.5) together with the corresponding AMPL data file and optimal solutions of them. For simplification and to save space, when considering the data file, we will only show how the paths are set for each demand because the initial parameters are not changed and only the number of candidate paths and the specific paths for each node are changed for every topology. Certainly, the number of paths for each demand will also change but one may figure it out through how the paths are set in the following.



(a) First topology case

```

set Path[1,1] := 1;
set Path[2,1] := 1;
set Path[3,1] := 1;
set Path[4,1] := 1;
set Path[5,1] := 1;
set Path[6,1] := 1;
set Path[7,1] := 1;
set Path[8,1] := 1;
set Path[9,1] := 1;
set Path[10,1] := 1;
set Path[11,1] := 1;

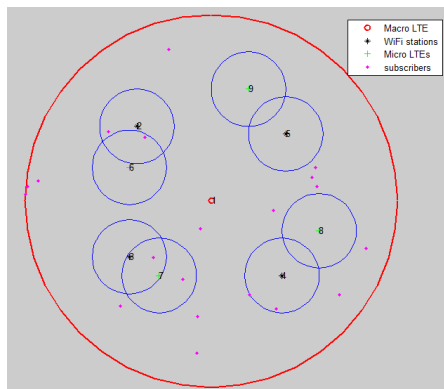
set Path[12,1] := 1; set Path[12,2]=3;
set Path[13,1] := 1; set Path[13,2]=3;
set Path[14,1] := 1; set Path[14,2] := 6;
set Path[15,1] := 1; set Path[15,2] := 4;
set Path[16,1] := 1; set Path[16,2] := 4;

set Path[17,1] := 1; set Path[17,2] := 8;
set Path[18,1] := 1; set Path[18,2] := 8;
set Path[19,1] := 1; set Path[19,2] := 2; set Path[19,3]=6;
set Path[20,1]=1; set Path[20,2]=3; set Path[20,3]=7;

```

(b) The specific data file

Figure 4.5: First topology: 11 demands only have access to macro-LTE whereas 9 nodes have two or more than two ways to access the service. Note that we treated the node on the boarder of a coverage as inside that. The final allocation sorted in non-decreasing order is  $\mathbf{h} = (9.09, \dots, 9.09, 9.09, 50, 50, 100, 166.667, 166.667, 166.667, 200, 200, 400)$ .



(a) Second topology case

```

set Path[1,1] := 1;
set Path[2,1] := 1;
set Path[3,1] := 1;
set Path[4,1] := 1;
set Path[5,1] := 1;
set Path[6,1] := 1;
set Path[7,1] := 1;
set Path[8,1] := 1;
set Path[9,1] := 1;
set Path[10,1] := 1;
set Path[11,1] := 1;
set Path[12,1] := 1;
set Path[13,1] := 1;
set Path[14,1] := 1;

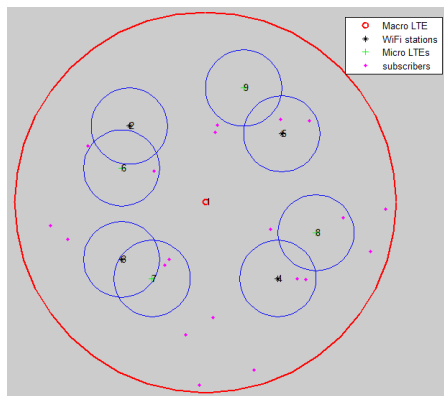
set Path[15,1] := 1;set Path[15,2] := 4;
set Path[16,1] := 1;set Path[16,2] := 4;

set Path[17,1] := 1;set Path[17,2] := 2;
set Path[18,1] := 1;set Path[18,2] := 7;
set Path[19,1] := 1;set Path[19,2] := 2;set Path[19,3]=6;
set Path[20,1]:=1;set Path[20,2]:=3;set Path[20,3]:=7;

```

(b) The specific data file

Figure 4.6: Second topology: 14 demands only have access to macro-LTE whereas 6 nodes have two or more than two ways to access the service. Note that we treated the node on the boarder of a coverage as inside that. The final allocation sorted in non-decreasing order is  $\mathbf{h} = (7.142, \dots, 7.142, 7.142, 100, 200, 200, 250, 250, 400)$ .



(a) Third topology case

```

set Path[1,1] := 1;
set Path[2,1] := 1;
set Path[3,1] := 1;
set Path[4,1] := 1;
set Path[5,1] := 1;
set Path[6,1] := 1;
set Path[7,1] := 1;
set Path[8,1] := 1;
set Path[9,1] := 1;
set Path[10,1] := 1;
set Path[11,1] := 1;

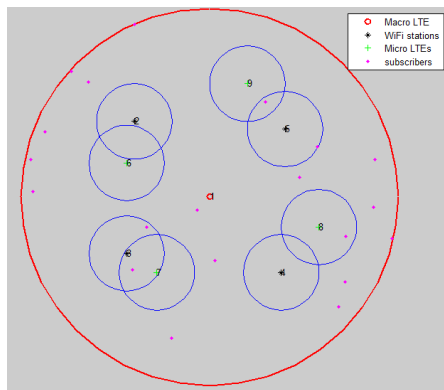
set Path[12,1] := 1;
set Path[13,1] := 1; set Path[13,2]=8;
set Path[14,1] := 1;set Path[14,2] := 6;
set Path[15,1] := 1;set Path[15,2] := 5;
set Path[16,1] := 1;set Path[16,2] := 5;

set Path[17,1] := 1;set Path[17,2] := 4;
set Path[18,1] := 1;set Path[18,2] := 4;
set Path[19,1] := 1;set Path[19,2] := 7;
set Path[20,1]:=1;set Path[20,2]:=7;

```

(b) The specific data file

Figure 4.7: Third topology: 12 demands only have access to macro-LTE whereas 8 nodes have two or more than two ways to access the service. The final allocation sorted in non-decreasing order is  $\mathbf{h} = (8.333, \dots, 8.333, 8.333, 50, 50, 100, 100, 200, 200, 200, 200)$ .



(a) Fourth topology case

```

set Path[1,1] := 1;
set Path[2,1] := 1;
set Path[3,1] := 1;
set Path[4,1] := 1;
set Path[5,1] := 1;
set Path[6,1] := 1;
set Path[7,1] := 1;
set Path[8,1] := 1;
set Path[9,1] := 1;
set Path[10,1] := 1;
set Path[11,1] := 1;
set Path[12,1] := 1;
set Path[13,1] := 1;
set Path[14,1] := 1;

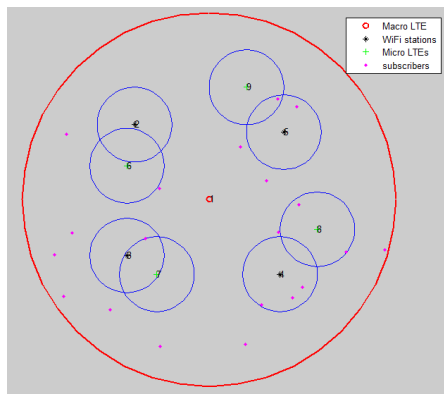
set Path[15,1] := 1;
set Path[16,1] := 1;set Path[16,2] := 3;

set Path[17,1] := 1;set Path[17,2] := 5;
set Path[18,1] := 1;set Path[18,2] := 8;
set Path[19,1] := 1;set Path[19,2] := 5;set Path[19,3]=9;
set Path[20,1]:=1;set Path[20,2]:=3;set Path[20,3]:=7;

```

(b) The specific data file

Figure 4.8: Fourth topology: 15 demands only have access to macro-LTE whereas 5 nodes have two or more than two ways to access the service. The final allocation sorted in non-decreasing order is  $\mathbf{h} = (6.667, \dots, 6.667, 6.667, 100, 250, 250, 250, 250)$ .



(a) Fifth topology case

```

set Path[1,1] := 1;
set Path[2,1] := 1;
set Path[3,1] := 1;
set Path[4,1] := 1;
set Path[5,1] := 1;
set Path[6,1] := 1;
set Path[7,1] := 1;
set Path[8,1] := 1;
set Path[9,1] := 1;
set Path[10,1] := 1;
set Path[11,1] := 1;

set Path[12,1] := 1; set Path[12,2]=5;
set Path[13,1] := 1; set Path[13,2]=4;
set Path[14,1] := 1;set Path[14,2] := 4;
set Path[15,1] := 1;set Path[15,2] := 4;
set Path[16,1] := 1;set Path[16,2] := 8;

set Path[17,1] := 1;set Path[17,2] := 8;
set Path[18,1] := 1;set Path[18,2] := 8;
set Path[19,1] := 1;set Path[19,2] := 5;set Path[19,3]=9;
set Path[20,1]:=1;set Path[20,2]:=3;set Path[20,3]:=7;

```

(b) The specific data file

Figure 4.9: Fifth topology: 11 demands only have access to macro-LTE whereas 9 nodes have two or more than two ways to access the service. The final allocation sorted in non-decreasing order is  $\mathbf{h} = (9.09, \dots, 9.09, 9.09, 33.333, 33.333, 33.333, 133.333, 133.333, 133.333, 250, 250, 500)$ .

The optimal solution is below:

```
h [*] :=
1  9.09091
2  9.09091
3  9.09091
4  9.09091
5  9.09091
6  9.09091
7  9.09091
8  9.09091
9  9.09091
10 9.09091
11 9.09091
12 166.667
13 166.667
14 100
15 200
16 200
17 50
18 50
19 400
20 166.667
;
```

(a) AMPL result of the first topology

The optimal solution is below:

```
h [*] :=
1  7.14286
2  7.14286
3  7.14286
4  7.14286
5  7.14286
6  7.14286
7  7.14286
8  7.14286
9  7.14286
10 7.14286
11 7.14286
12 7.14286
13 7.14286
14 7.14286
15 200
16 200
17 250
18 100
19 250
20 400
;
```

(b) AMPL result of the second topology

The optimal solution is below:

```
h [*] :=
1  8.33333
2  8.33333
3  8.33333
4  8.33333
5  8.33333
6  8.33333
7  8.33333
8  8.33333
9  8.33333
10 8.33333
11 8.33333
12 8.33333
13 100
14 100
15 200
16 200
17 200
18 200
19 50
20 50
;
```

(c) AMPL result of the third topology

The optimal solution is below:

```
h [*] :=
1  6.66667
2  6.66667
3  6.66667
4  6.66667
5  6.66667
6  6.66667
7  6.66667
8  6.66667
9  6.66667
10 6.66667
11 6.66667
12 6.66667
13 6.66667
14 6.66667
15 6.66667
16 250
17 250
18 100
19 250
20 250
;
```

(d) AMPL result of the fourth topology

The optimal solution is below:

```
h [*] :=
1  9.09091
2  9.09091
3  9.09091
4  9.09091
5  9.09091
6  9.09091
7  9.09091
8  9.09091
9  9.09091
10 9.09091
11 9.09091
12 250
13 133.333
14 133.333
15 133.333
16 33.3333
17 33.3333
18 33.3333
19 250
20 500
;
```

(e) AMPL result of the fifth topology

Figure 4.10: The AMPL results of the above topologies

In order to make it clear, Figure 4.10 has shown the actual results in AMPL of the above topologies, although we have given in the figures' descriptions.

So far, the illustration of the numerical results has come to an end. As we mentioned, the five topologies are very typical ones and most of others are quite similar. For example, in the fifth case, more than 9 demands have multiple access to the base stations. Without AMPL, it could be very complicated to compute the total allocation vector. In reality, a particular area may hold more than hundreds of people; if 30 of them have multiple access, this can be impossible to solve by hand.

### 4.3 Comparison of Three Schemes

To have more advanced results and to show the advantage of our proposed Max-min algorithm, in this section, we will compare its performance with two other traditional heuristic schemes. The first is called *WiFi-preferred* scheme, a non-bifurcated case, where a subscriber chooses the RAT based on the *capacity-to-coverage* ratio in decreasing order. That is to say that WiFi RAT is chosen with priority since it has a higher capacity when a user is within the coverage of WiFi. If a user has problem with connecting the WiFi, the nearest pico cell is selected and then macro-LTE. Basically, the objective of this scheme is to offload the more expensive cellular network maximally.[4]

The other conventional scheme for comparison is *Max-usage* or "greedy" scheme. In contrast, it is a bifurcated scheme and moreover, a user-centric scheme. In this case, users are able to transmit on all available radio interfaces simultaneously. Apparently, this scheme is power-hungry but it can significantly improve the throughput in comparison with the above one. More details of WiFi-preferred and Max-usage schemes can be found in [19] and [20].

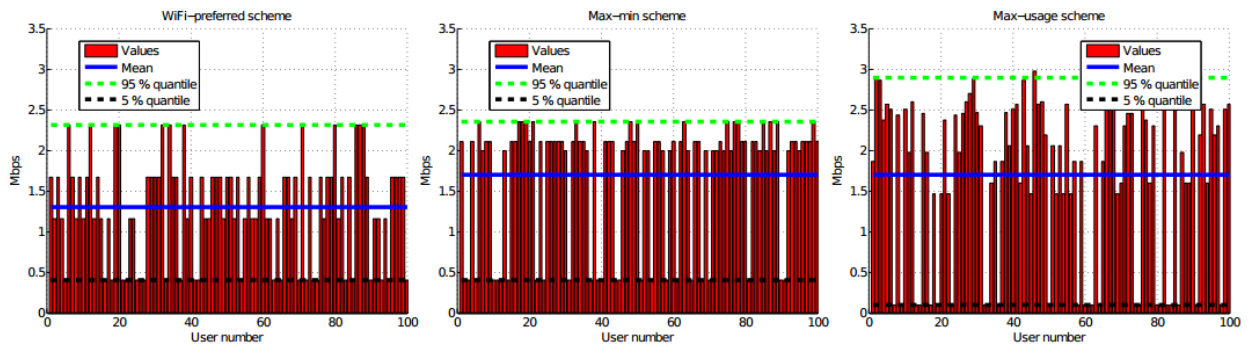


Figure 4.11: Per-user throughput performance comparison: (a) WiFi-preferred, (b) Proposed Max-min, and (c) Max-usage schemes [4]

To simulate a more challenging realistic deployment, we established a scenario with 100 subscribers, where 25% can only connect to macro-LTE, other 50% have

access to either micro-LTE or WiFi or both of them and the remaining 25% are free to choose between three of them. There are still one micro cell, four micro-LTEs and four WiFis and the throughput for them is: 10 Mbps (for both macro- and micro-LTE) and 30 Mbps for WiFi. [4]

Figure 4.11 shows the results of the comparison of performance with throughput in the mean, 5%-quantile and 95%-quantile. What is concluded is that in terms of the average peruser throughput, our proposed scheme Max-min algorithm in subfigure (b) is higher than that of Wifi-preferred scheme. Further, although Max-min and Max-usage schemes are sharing the same value of "mean" quantile, the throughput of the users served by macro-LTE only is dramatically improved (see the 5% quantile line). Therefore, Max-min algorithm not only combines the benefits but also avoids shortcomings of both counterpart heuristic schemes. [4]

## 5. CONCLUSION

The objective of the thesis was to solve N-tier heterogeneous network resource allocation problem through considering it as a network flow problem. The results have turned out that formulating a HetNet problem as a NFP is an effective way and the research has proposed a new method for assisted network selection.

In the above chapters, the author has introduced:

- the fundamental and core notions and notations of network flow problem.
- the fair network concept and the corresponding max-min fairness (MMF) criterion.
- the formulation of the targeted 3-tier HetNet network problem.
- two available algorithms: one is based on NBT-1 and the other on NBT-3.
- a brief and specific introduction to AMPL language with an understandable example; how the concentrated problem is established in AMPL and how AMPL has solved it with CPLEX.
- illustration to the topologies set by MATLAB and the corresponding results to 5 of them.
- comparison with two counterpart heuristic schemes: WiFi-preferred and Max-usage.

The general findings of the thesis are as follows. Firstly, MMF is the most fair way for resource or bandwidth allocation among users. For our particular problem, the traffic demands (subscribers) are set to have one or more paths for traffic realizing. Based on MMF criterion, algorithm (3.2.1) or algorithm (3.2.2) which we adopted has accurately solve the fair allocation between users. Secondly, AMPL modeling software is an excellent tool for modeling our network problem especially when we used algorithm (3.2.2) as it is actually a typical linear program problem. Since dual variables have played a very important role in achieving the solution, AMPL language together with CPLEX solver have applied dual theory very well. As we mentioned, at first we adopted solver MINOS which has made many errors. The efficiency of AMPL tool is rationally high. For a 3-tier HetNet network with 20 subscribers and 9 base stations, it can solve within 5 seconds. Thirdly, the topologies we

built are relatively reasonable. They are able to cover most locations of a subscriber when the position of a BS is determined. Fourthly, the final total demand volume allocation is lexicographically maximal which means that each demand has obtained the optimal allocation solution and it is the most fair one. Fifthly, our proposed algorithm—Max-min fairness is an excellent scheme for the typical heterogeneous networks currently. It not only maintains a fairly good average performance, but also delivers the maximum fairness to the system.

However, there are some insufficiencies about the topologies. The one thing is that in terms of the number of subscribers, the coverage of macro-LTE in our case is relatively a bit large, which leads to some situation where almost all subscribers are located outside either WiFi stations or micro-LTE stations and only a few (1 or 2 are within either BS). In this case, there is no meaning to study since we can simply compute it by hand. Figure 5.1 is one of these examples. It is so simple that the

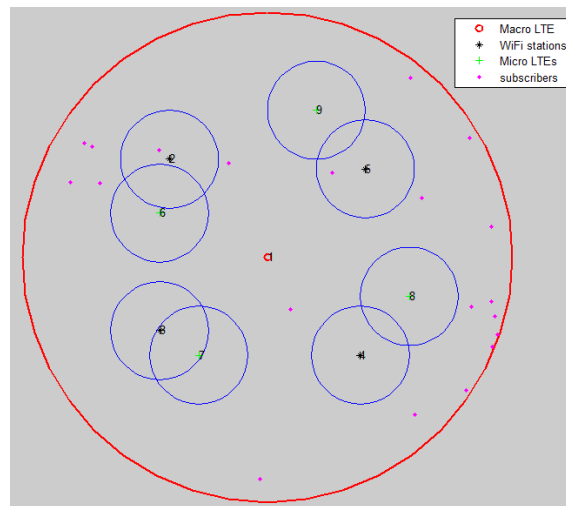


Figure 5.1: A situation where most subscribers are only served by macro-LTE station

optimal solution is to distribute equally for those that are only served by macro-LTE and the single subscriber within either WiFi or micro-LTE coverage gains the whole resource or bandwidth. Therefore, this is basically what needs to be improved.

For further study of this subject, one can create a more realistic case: since in reality, WiFi or micro-LTE stations will mostly concentrate on places where hold a high visitor flow rate, thus, the situation gets more complicated because more subscribers may have multiple access to wireless resources.

In summary, this chapter has reviewed what has been done in this thesis, found what needs to be further improved and given the suggestion for future study. The idea to solve HetNet network problem by NFP has filled the gap in this area and as the increasing interest in this subject, more advanced techniques can be developed.

## REFERENCES

- [1] J. Andrews, "Seven ways that HetNets are a cellular paradigm shift," *IEEE Comm. Mag.*, vol. 51, 2013.
- [2] J. G. Andrews et al., "Femtocells: Past, Present, and Future," *IEEE JSAC*, Apr. 2012.
- [3] D. P. Malladi, "Heterogeneous Networks in 3G and 4G," *IEEE Commun. Theory Wksp.*, <http://www.ieeeectw.org/program.html>, May 2012.
- [4] D. Moltchanov et al., (2014) "On the Optimal Assisted Rate Allocation in N-Tier Multi-RAT Heterogeneous Networks", *IEEE PIMRC 2014*, USA.
- [5] *3GPP TR 37.842, 3GPP/WLAN RAN Interworking Study Item Report*, 2013.
- [6] Vangie Beal, "What is WiFi", [http://www.webopedia.com/TERM/W/Wi\\_Fi.html](http://www.webopedia.com/TERM/W/Wi_Fi.html), May, 2014.
- [7] H. Dhillon et al., "Modeling and analysis of K-tier downlink heterogeneous cellular networks," *IEEE J. on Sel. Areas in Comm.*, vol. 30, pp. 550-560, 2012.
- [8] J. Andrews et al., "An overview of load balancing in HetNets: Old myths and open problems," *submitted to IEEE Wireless Comm.*, vol. TBD, p. TBD, 2013.
- [9] M. Pioro and D. Mehdi. *Routing, Flow, and Capacity Design in Communication and Computer Networks* (The Morgan Kaufmann Series in Networking). 1 ed., 2004.
- [10] L. Massoulié and J. Roberts, "Bandwidth sharing: objectives and algorithms," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1395 - 1403, 1999.
- [11] R. Fourer, D.M. Gay and B.W. Kernighan (2003). *AMPL: A Modeling Language for Mathematical Programming*, 2nd Edition, United States of America.
- [12] T. Cinkler, P. Laborczi and M. Pioro, "Fairness considerations with algorithms for elastic traffic routing" in *J. Telecomm. and Inf. Tech.*, vol. 2, pp. 1-10, 2004.
- [13] C. Johnson, *Long Term Evolution in bullets* (2nd Edition). CreateSpace, 2012.
- [14] "Study on WLAN/3GPP Radio Interworking," *3GPP Technical Report (TR) 37.834*, 2013.
- [15] H. Holma and A. Toskala, *WCDMA for UMTS: HSPA Evolution and LTE*. 5 ed., 2010.

- [16] "Access to the 3GPP Evolved Packet Core (EPC) via non-3GPP access networks," *3GPP Technical specification (TS) 24.302*, 2013.
- [17] S.Andreev et al., "Cellular traffic offloading onto network-assisted device-to-device connections," *IEEE Comm. Mag.*, vol. 52, pp.20-31, 2014.
- [18] "Further advancements for E-UTRA physical layer aspects," *3GPP Technical Report (TR) 36.814*, 2010.
- [19] N.Himayat, S.-P.Yeh, A.Y.Panah, S.Talwar, M.Gerasimenko, S.Andreev, and Y.Koucheryavy, "Multi-Radio Heterogeneous Networks: Architectures and Performance," in *Proc. of IEEE International Conference on Computing, Networking and Communications (ICNC)*, 2013.
- [20] "Guidelines for evaluation of radio interface technologies for IMTAdvanced," ITU, 2009.
- [21] M.Gerasimenko, N.Himayat, S.-p. Yeh,S.Talwar, S.Andreev, and Y.Koucheryavy, "Characterizing Performance of Load-Aware Network Selection in Multi-Radio ( WiFi / LTE ) Heterogeneous Networks," in *GLOBECOM Workshops (GC Wkshps)*, 2013.