Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

A local search approach to resolving capacity issues in mobile cellular networks

A thesis submitted to the
Delft Institute of Applied Mathematics
in partial fulfillment of the requirements

for the degree

MASTER OF SCIENCE
in
APPLIED MATHEMATICS

by

Sander Gribling
Delft, the Netherlands
May 2015

MSc THESIS APPLIED MATHEMATICS

"A local search approach to resolving capacity issues in mobile cellular networks"

Sander Gribling

Delft University of Technology

Daily supervisor

T. Ouboter, MSc

Responsible professor

Prof. dr. ir. K. I. Aardal

Other thesis committee members

Dr. D. C. Gijswijt

Prof. dr. J. L. van den Berg

May 2015

Delft, the Netherlands

Contents

Preface

Dear reader,

What lies before you is the result of my graduation work for the master Applied Mathematics. I have had the pleasure of doing my thesis at the 'Nederlandse Organisatie voor Toegepast-Natuurwetenschappelijk Onderzoek', or as most of you know it: TNO. The topic of my thesis is a decision support system for mobile cellular networks. This is part of the SEMAFOUR project TNO is involved in. The decision support system is designed to help network operators choose upgrades to the network based on predicted future bottlenecks. The recommendation to the network operator we provide in this thesis work is based upon a local search method. In fact, two different local search methods are compared. A large part of this thesis focuses on the choice of the initial state. It turns out that the computation time required by the local search methods decreases (a lot) by choosing the initial state wisely.

Besides working on my thesis project I have also contributed to the SEMAFOUR project in other ways. The internal deliverable 5.3 contains a general description, and some numerical results, of my project. It was nice to be involved in the process of writing and rewriting (based on external reviews). I have also contributed to the demonstrator of the decision support system. For the demonstrator, I was involved in making the demonstration scenario. My local search approach is used to create different solutions for the bottlenecks that arise in the demonstration scenario. The result of this work can be found here: [9].

A second side project to my thesis project is the development of a network evaluation tool. In Chapter 3 I have described the previously existing evaluation tool used in this project (SONlab). In a similar fashion I have presented my own evaluation tool in Appendix D.

I would also like to take the opportunity to thank all my supervisors. First of all Tanneke, my daily supervisor, thank you for forcing me to work in a more structured way (that is, to put my plans on paper). It might seem like a small thing, but it was really useful. Karen, Dion and Hans, thank you for all the feedback on my thesis. It has been really useful to get input from two different fields. Explaining the 'other' field to each of you helped me to understand it better and to write it down in an accessible way. Finally, Remco, thank you for your comments on the third chapter and all other small questions you answered about cellular networks.

A final word to the reader: I hope you enjoy reading it as much as I enjoyed writing it!

Sander Gribling, May 2015

Chapter 1

Introduction

We all have one, a mobile phone. What most of us do not realize is how complex a mobile network can be, and, how difficult it is to plan a good mobile network. For instance how should a network operator best adjust the network based on predicted traffic growth in certain areas? First of all, this is not an easy question! Even to just understand the question we would need to know what 'best' means. Most of us have used a mobile phone to connect to the internet, be it via WiFi or 3G or perhaps even through a 4G network. Most likely you have at some point experienced a 'bad' connection, for instance when looking at a video on internet it paused to buffer. A network operator wants to be able to guarantee its users a certain performance of its network, say for instance that 99% of the time you can achieve a data connection of 0.5 Mbps. However it is also a for-profit company, so it wants to keep the cost as low as possible. A trade-off between these two goals needs to be made. We will develop a method to give a network operator advise on how to upgrade its network in case of a predicted future performance bottleneck in such a way that the network will continue to meet its performance standards and that it is a cost-efficient solution. Nowadays, most of the above is done manually, based on human experiences. The method we have in mind is highly automated. It combines input about demand and performance forecasts into a list of promising network upgrades automatically.

In this chapter we would like to cover two subjects:

1. An introduction to mobile networks
2. The problem description

A mobile cellular network comes with a lot of terminology, a detailed description of this terminology would become quite lengthy. That is why we introduce the concept of such a network using a few sketches. A detailed description of a mobile network can be found in Appendix A. These sketches should be sufficient to understand the basic concepts used in this thesis, in particular the problem definition. This thesis is carried out at TNO as a part of the European SEMAFOUR project and therefore we briefly introduce the SEMAFOUR project and its goals. From that, our problem definition follows naturally. After defining the problem we review the literature related to this subject and finally we give an outline of our approach to solving the problem.

1.1 A mobile cellular network

Figure 1.1: Sketch of a mobile cellular network.

Let us start by giving a general description of a mobile network. In Figure 1.1 we can see a sketch of a mobile network. The figure shows us three large antennas, four small antennas, and several types of users (mobile phone, laptop, someone in a car, a household, an office building). Each user connects to an antenna and the area in which users connect to a certain antenna is shown as a circle around the antenna. Of course in reality these areas are not perfect circles, in a moment we will explain what influences their shape. First let us look at a side view of an antenna and a user, see Figure 1.2. What we see is on the left an antenna and on the right a user, the cone drawn from the antenna to an area around the user represents the 3D area for which the antenna could provide a user with a connection. Some terminology is also introduced; a user is often referred to as User Equipment, UE, this allows us to not distinguish between mobile phones, laptops, etc. In fact, we do not model individual users, we mostly consider area based statistics and as such we look at Service Test Points, STPs, which are small regions with a certain traffic intensity.

As we can see from Figure 1.2 an antenna is aimed in a certain direction, this allows us to place multiple antennas on a single tower. Such a collection of antenna at one location is referred to as a base station, BS. In Figure 1.3 we can see a base station and its service areas. These service areas (one for each antenna) are usually referred to as either cells or sectors. Figure 1.3 shows us a 3-sector base station along with the resulting cell structure. We have indicated again our Service Test Points, a grid of squares. In Figure 1.3 we have sketched three cells, therefore our base station has a 3-sector configuration (three antennas). The cells do not have a nice geometric shape in our sketch, this is because in reality they also do not have a nice shape. The assignment of an STP to a cell is done based on the best server principle, where the best server is defined as the antenna which provides the strongest pilot signal (a signal broadcast at a specific frequency by all antennas at a fixed power setting). The signal strength depends on path loss which is not uniform due to obstacles between the antenna and UE.

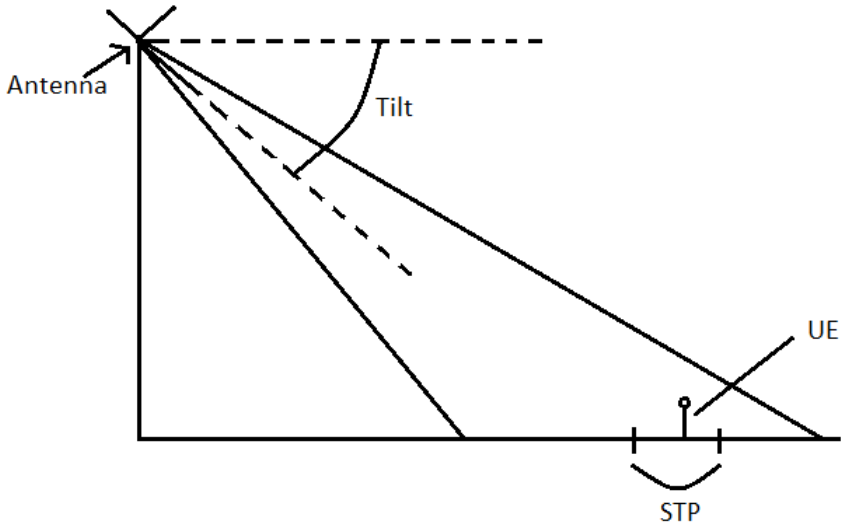

Figure 1.2: Sketch of an antenna and a mobile user (User Equipment, UE), side view.

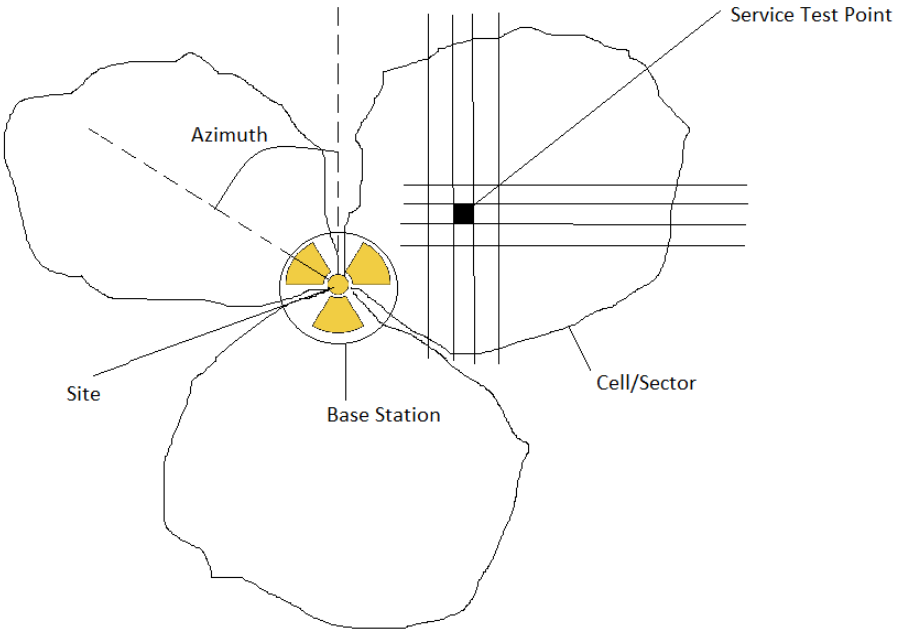

Figure 1.3: Sketch of a top-down view of a base station with three sectors/cells and service test points.

The other concepts that can be seen in Figures 1.2 and 1.3 are tilt and azimuth. An antenna is aimed in a certain direction, this direction is specified by the offset to a certain reference direction. The offset in the vertical plane is called tilt (reference: the horizontal direction), the offset in the horizontal plane is called azimuth.

Another aspect of a mobile network is its performance. We use simulators to evaluate the performance of a network configuration. The performance of a network is hard to capture in one figure since we have multiple stakeholders, each with a different view of what a good performance is. Important examples of stakeholders are the network operator and the users. Users can also be divided into several categories, for example slow and fast moving users. Fast moving users would value a fast and good transition from one cell to another (that is, a handover), whilst slow moving users might value this as less important. It is for this reason that we describe the performance of a network by certain Key Performance Indicators (KPI) . Therefore, an evaluation of the performance of a network configuration comes down to calculating the KPIs.

An important KPI we use is the load vector. The load per cell can be seen as the ratio between the physical resources requested by the users connected to that cell and the total available resources of that cell. The total amount of available resources is called the available bandwidth or spectrum. The available spectrum depends on what the operator has bought at the auction, a typical spectrum would be the range between 1800 and 1810 MHz. The load vector is an important measure of the networks performance as it relates to user experience (for example calls being blocked or a lower data throughput received than requested). In Appendix A.2 we have defined the load and other KPIs that we are interested in.

As we have mentioned above we want to evaluate the performance of a network configuration. Since we use it often it is important to clarify our definition of a network configuration.

Informal definition 1.1.1 (Network configuration)**.** A network configuration consists of a set of active base stations. Each base station has a location and a number of antenna arrays. Each antenna array is of a certain type and has parameters transmission power, tilt and azimuth.

The reader who is unfamiliar with any of the above terms we would like to refer to Appendix A.

1.2 The SEMAFOUR project

This project is carried out at TNO as a part of the European SEMAFOUR project, which stands for Self-Management for Unified Heterogeneous Radio Access Networks. The SEMAFOUR project is carried out by a consortium of companies namely: Nokia, atesio, Ericsson, iMinds, France Télécom - Orange labs, Telefónica I+D, TNO and the Technische Universität Braunschweig. It is partly funded through the European Commission via the funding scheme FP7 STREP. The goals of the SEMAFOUR project are best described on the projects website [2]. The following (in italics) is a word-for-word copy of the projects general goals.

The SEMAFOUR project will design and develop a unified self-management system, which enables the network operators to holistically manage and operate their complex heterogeneous mobile networks. The ultimate goal is to create a management system that enables an enhanced quality of user experience, improved network performance, improved manageability, and reduced operational costs.

In other words, the goal of the SEMAFOUR project is as follows: to create a system which is able to automatically manage the mobile network of a network operator. By automatic management we mean that the system should be able to detect aspects of the network which can be improved and provide the network operator with options to achieve such an improvement. Some of these options could be implemented without human intervention (for example the fine tuning of power settings of the antennas to minimize interference), others require human interaction. The long term planning, i.e., building new sites, usually requires some human interaction. The Decision Support System (DSS) is part of this long term planning. The aim of the DSS is to automatically give recommendations about network upgrades based on predictions about future bottlenecks. The recommended upgrades should make sure that the network will meet the operators performance standards for the at least the time frame for which the prediction was made. The DSS consists of two parts, bottleneck detection and recommendations for network upgrades. The first part of the DSS consists of detecting bottlenecks based on predictions about future traffic. The output of this first part will be the type of bottleneck and the relevant problem area. These will serve as input for the second part. The output of the second part will be a list of promising upgrades the operator can choose from to prevent the predicted future bottleneck. This thesis focuses on the second part.

The second part of the DSS should give recommendations about network upgrades. One can distinguish two different classes of upgrades: expanding capacity using the same radio access technology or migrating to a new technology. These two classes are treated separately within the SEMAFOUR project in the following use cases (in the same order):

- Operational Network Evolution (DSS-ONE),
- Strategic Network Migration (DSS-SNM).

This thesis is largely contained in the first use case. Given this classification we can visualize the DSS in the diagram shown in Figure 1.4.

Another part of the SEMAFOUR project that we would like to mention is the following objective of SEMAFOUR: *To develop a demonstrator that proves, through simulation in realistic scenarios, that the vision, concepts, methods and algorithms developed within the project provide significant benefits for management and operation and result in performance improvements. A key target of this demonstrator is to provide an intuitive visualisation of the benefits in terms of performance, efficiency and flexibility enhancements achieved by a unified self-management system for multi-RAT, multi-layer radio access networks.*

Some of the terms we see in this objective have not been explained before. RAT stands for Radio Access Technology. LTE, HSPA and WLAN are examples of these kinds of technologies[1]. In this thesis we consider LTE based networks.

[1]The existence of different technologies suggests that they behave in different ways. This is

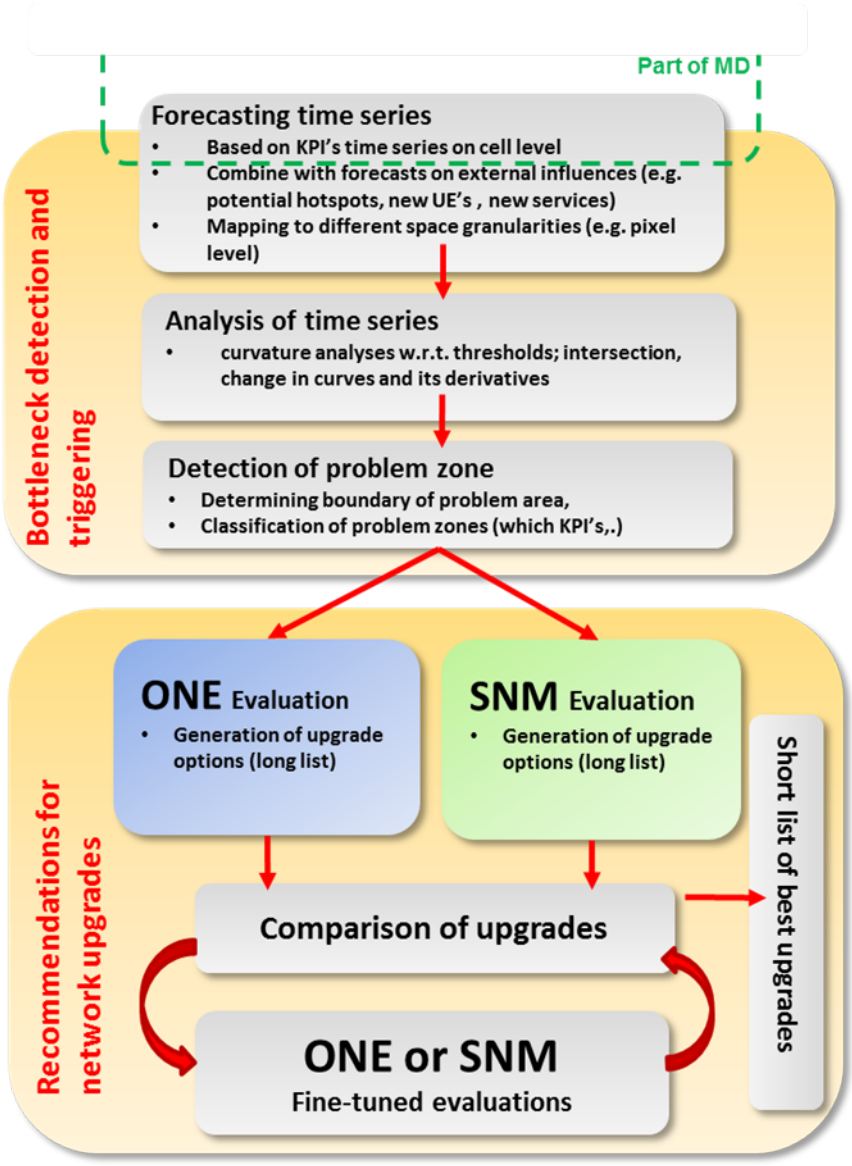

Figure 1.4: Schematic representation of the DSS.

LTE stands for Long Term Evolution. The reason we mention this objective of the SEMAFOUR project is also mentioned in the preface: the author of this thesis has played a role in the development of one of the demonstrator scenarios, the one relating to the subject of this thesis, and the generation of all

indeed the case. The main difference is in the way in which they assign the available resources at a base station (bandwith, i.e. hertz) to users. In particular inter- and intra-cell interference depend on the way the resources are assigned to users. LTE is the latest technology, it eliminates intra-cell interference by using the radio access technology Orthogonal Frequency Division Multiple Access (OFDMA). Further information about LTE or OFDMA can be found in [28].

data related to that scenario.

1.3 Problem description

As can be seen in the previous section the goal of this thesis is to provide recommendations of network upgrades needed to prevent predicted future performance bottlenecks. To be clear: the prediction of the future bottlenecks is not part of this thesis, it has already been done within the SEMAFOUR project. Before going into any more details about the type of upgrades available we can already state the problem definition:

What is 'the best way' to alter the existing mobile cellular network in order to prevent a predicted future bottleneck?

'The best way' depends on what the network operators goals are. The network operator would most likely want to have a cost efficient network (upgrade) providing a good (or the best possible) Quality of Service (QoS) or Quality of Experience (QoE) level. The difference between QoS and QoE being that the first is an objective measure and the second a much more subjective one. To give an idea of how a network performs on QoS and QoE levels there are certain Key Performance Indicators (KPIs) which can be looked at. In Section 1.1 we introduced the concept of load. Load is a very important indicator since a high load on certain cells is a strong indicator of an area that has a reduced QoS. A further motivation to use load as our KPI is that other important KPIs such as user data, throughput and blocking rate (the probability that there are not enough resources available for a user) are strongly connected to the load of the network.

We see that the goals of a network operator are conflicting in the sense that a network which provides a high QoS level will most likely not be cheap and the other way around. The weighing of performance indicators and costs of the network (upgrade) is therefore something to which we need to pay special attention. The weights should be determined by the operator as it reflects the priorities they give to the KPIs.

Lastly we note that the first part of the DSS categorizes the bottlenecks according to the following problem types:

1. Not enough coverage
2. Not enough capacity

In this thesis we consider problems arising in an existing network, coverage should therefor not be an issue, so we consider problems of the second type.

To complete the problem description we still need to define what kind of network upgrades we consider. First of all we note that we do not do the fine tuning of the base stations, i.e., given a selection of base stations we do not attempt to find the best possible configuration for each base station in terms of power, tilt, azimuth, etc. This is because we do long term planning and the configuration typically changes within short time spans. Hence, we assume that for each base station we are given a configuration that is close to optimal, or at least a reasonable choice for that location. The types of base stations

we consider are macro sites with three or six sectors and micro sites with one sector. The macro and micro says something about the area they are designed to cover. A macro site is typically placed at a high position and has a high power usage (around 46 dB), so that it can serve a large area (observed range is around 800m). Micro sites however are typically placed at a hight of around 30m and with a lower power usage (30 dB) than the macro sites, their observed range is more around 100 to 200 meters. We consider two possible upgrades to the network:

1. The 6-sectorization of a 3-sector site. Which means that we replace the 3-sector BS by the 6-sector BS at the same site.

2. The placement of a new micro site.

We would like to point out that although we only consider the case with two possible upgrade types the methods we will develop (in particular the local search heuristic) can be easily adapted to use more upgrade options. We assume that a list of potential locations for micro sites is provided.This reflects reality well in the sense that you simply can not build a new site on every location (terrain restrictions, permits, etc.). It is therefore acceptable to assume that the network operator provides a finite list of potential locations for micro sites.

A concise formulation of the problem would be the following:
Given a prediction about the future traffic demand and a list of potential network upgrades, select those upgrades that make the network meet the operators KPI requirements for the predicted traffic state, at minimum cost.

1.3.1 Related literature

The problem as we have defined it is often called the antenna placement problem (APP). Some variations of the problem have already been considered. In most of the available literature the radio access technology considered was different from LTE (usually GSM or UMTS was used). When reading articles where a simplification of the behavior of a mobile network was made, we need to keep in mind that LTE has different properties than for instance GSM or UMTS, in particular when simplifications of the interference are made. The complexity of the APP is known to be NP-hard[2], in [43] it is studied in an accessible way. They simplify the APP by ignoring interference and instead assume that the resource demand of a user to a BS is independent of the load. The possible assignment of a user to a BS only depends on path loss and the BSs configurations. They then relate this simplification to the problem of finding a minimum (size) dominating set. The Dominating Set problem is shown to be NP-hard in [13], it is even hard to approximate (no polytime[3] algorithm exists with approximation ratio $c \log n$ for any $0 < c < 1/4$).

Since even approximating this simplification cannot be done in polynomial time it seems natural to consider heuristic methods. To the reader who is not familiar with heuristic methods: Chapter 2, introduces the methods mentioned below.

[2]See Chapter 2 for an explanation of the concept NP-hard.
[3]See Chapter 2 for an explanation of the concept polytime.

First of all [42] provides an extensive overview of the work that has already been done on the antenna placement problem, often heuristic approaches are used. In particular the authors mention that the APP is a special case of the well known facilities location problem. In [20], [21] and [43] the APP is formulated as a mixed integer program (slight variations between authors exist). Tabu search is used by [15], [38], [20], and [21]. St-Hilaire et al., [38], is noteworthy since they split the evaluation of a network configuration into multiple phases, some of which can be solved to optimality very fast, considerably improving the computation time needed for an evaluation. The most basic local search method used is Greedy search, it is often compared to Simulated Annealing and/or genetic algorithms. Articles in which a comparison between Greedy search and Simulated Annealing is made are [10], [42], [32], and [36]. In [29] only Simulated Annealing is considered, they do however describe and explain the neighborhood structures they have used very thoroughly, it is for that reason that this article has formed the first inspiration for the local search method we have proposed in this thesis.

In [35] Tutschku introduces the concept of demand nodes. Demand nodes can be seen as a division of the problem area in smaller areas with a fixed demand. That means you do not look at individual users but for example at pixels. This is also a concept we will use. In [35] the concept of demand nodes is used to look at the coverage problem. Tutschku formulates this as the Maximal Coverage Location Problem. This model does not take capacity into account. In [37] the APP is considered for GSM networks. The authors propose an algorithm that could be used to optimize the configuration of each base station. It does so by considering each BS individually and assuming no other BSs are active, it then selects the settings which maximize coverage whilst meeting the capacity constraint. We consider the configuration of each base station as input, but [37] shows how it could be obtained in another way.

The articles mentioned above mostly use a single objective. In [41] the concept of dealing with multiple objectives is considered for the APP. Without presenting any comparative results they mention three possibilities:

1. Combine all objectives into a single scalar value, typically as a weighted sum, and optimize the scalar value.

2. Solve for the objectives in a hierarchical fashion, optimizing for a first objective then, if there is more than one solution, optimize these solution(s) for a second objective. Repeat until all objectives are considered.

3. Obtain a set of alternative, non-dominated solutions, each of which must be considered equivalent in the absence of further information regarding the relative importance of these objectives. Domination is in the Pareto optimality sense.

As we have mentioned before we consider a different radio access technology than what was used in the available literature, therefore we also need a different way to evaluate the performance of a network configuration. The doctoral work of Kimmo Hiltunen, [34], contains a lot of work done on performance evaluation of various network deployments. In particular simulators are considered along with a lot of parameters which could be used as a standard for various effects.

Seeing all of the work that has already been done it is natural to ask the following question: 'what does this thesis add to the existing knowledge?'. In previous work either a local search method or a mixed integer linear programming formulation was used. Never both. We do combine these methods and as such we get the best of both. Secondly, as far as we know, this is the first time that networks based on the LTE technology are considered. We give some insight into the network evaluation methods used, which take into account the specifics of the LTE technology.

1.3.2 Outline of the optimization approach

The problem as defined in Section 1.3 will be solved in the remainder of this thesis. As we have mentioned in Section 1.3.1 the problem is NP hard, even difficult to approximate, and has been studied by other authors using heuristic methods as well. We also use heuristics to 'solve'[4] the problem. In Chapter 2 we introduce the basic concepts of local search methods, along with a discussion of several well known variants. We primarily focus on the development of local search methods with a fast convergence (due to very specific neighborhoods), the reason we do so is that each evaluation of a network configuration is time expensive ($>$ 10 seconds). In Chapter 4 we specify our local search methods. A way to speed up convergence is to choose a good initial state. In Chapter 5 we discuss simplifications of the problem defined in Section 1.3 which can be solved fast and with a good initial state as outcome. The simplification we consider is a model where interference is not taken into account[5], formulated first as a binary integer program and later relaxed as a mixed integer linear program. In Chapter 6 we present our numerical results, based on a set of problem instances also defined in that chapter. Chapter 6 compares several local search methods on their performance. Since the initial state can be a big influence on the performance we see the initial state as a variable as well. We consider two initial states, the first being the existing network, the second the initial state as found using the methods of Chapter 5.

[4] In Chapter 2 we will note that heuristics attempt to solve a problem but are not always successful. Nonetheless, for ease of reading, we will say that a heuristic solves a problem when we actually mean that it attempts to solve the problem.

[5] In fact, it is guessed at the beginning, in the formulation of the program, independent of the actual interference resulting from the load of each cell.

Chapter 2

Introduction to mathematical optimization techniques

In this chapter we present an introduction to the mathematical techniques we use. First we explain the need for, and basic concepts of, local search methods. We then give a comparison of the benefits and downsides of a few of the most common local search methods (Greedy, Simulated Annealing and Tabu search). Secondly we give an introduction to ((mixed) integer) linear programming and relaxation techniques.

Before we continue, we note that the problem we are interested in has a natural formulation as a minimization problem: minimize the total costs whilst fulfilling some criterion (for example, the load of all cells must be below a certain threshold). In this chapter we therefore also formulate everything in terms of minimization problems. For example, when we compare two objective values we will say that the one with a lower value is better.

2.1 Computational complexity

As we have mentioned before, we use local search methods to solve the problem described in Section 1.3. In this sentence, 'solve' is a slightly misleading term since local search methods are heuristics. And heuristics in general do not actually solve the problem. A heuristic method attempts to find a very good solution using as much properties about the search space as possible, the obtained solution does not need to be optimal. Later on in this chapter we will see that the obtained solution is often a local optimum. That means that in a small region (neighborhood) around the obtained solution it is optimal. A local optimum which is also the best state in the search space is called a global optimum and an algorithm which is guaranteed to find a global optimum is called exact.

Heuristics in general do not provide a global optimum. Still they are sometimes used. The reason for this is that not all problems are equally difficult. For some problems exact algorithms require too much computation time. We want

to call certain types of problems 'difficult' and others 'easy'. One can imagine that even for difficult types of problems very small instances can be solved quite fast. So giving a bound of the form *too much computation time means more than a thousand calculations are needed* is not helpful (although in practice such bounds could determine your choice of algorithm). Instead we would like to say something about how quickly the amount of calculations needed grows when the instance size grows[1]. If this rate of growth is polynomial in the instance size we say that the algorithm used to solve this problem runs in polynomial time (and the problem is of the class P). Problems for which an exact algorithm exists which runs in polynomial time we call easy. Many mathematicians agree that not all problems can be solved in polynomial time. Another important class of problems is those problems for which a method exists to verify that a given solution is indeed an optimal solution, this class is called NP. The aforementioned belief of many mathematicians is that the classes P and NP are not the same. The most difficult problems in NP are called NP-complete and the problems which are at least as hard as any problem in NP are called NP-hard. For NP-hard problems we do not expect to be able to find an exact algorithm that runs in polynomial time.

In Section 1.3.1 we have referred to articles which say that the problem we posed in Section 1.3 is NP-hard. As such we do not expect to be able to find exact algorithms to solve it that run in polynomial time. It is for this reason that we consider a heuristic approach through local search methods.

2.2 Local search methods

In this section we explain what a local search method is. First of all let us consider the following general minimization problem:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & x \in U. \end{aligned} \tag{2.2.1}$$

where U is some subset of $\mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ is some function. The key characteristic of a local search method used to solve this problem is that it is an iterative procedure which starts with some $x_0 \in \mathbb{R}^n$ (not necessarily feasible) and in each iteration i attempts to move to an x_i with some relation to x_{i-1}; x_i has to be a neighbor of x_{i-1} (a proper definition of a neighbor will be given later on). The move is accepted if x_i is a 'better' solution than x_{i-1}, if the move is rejected we take $x_i = x_{i-1}$. The point x_i is called a state. This leads to the definition of a local search method:

Definition 2.2.1 (Local search method)**.** A local search method is an iterative procedure in which a neighbor of the current state is evaluated. If it meets the acceptance criterion it is accepted and we move to that state. If it is not accepted we do not change the current state. This process is repeated in the next iteration.

[1] As an example of an instance size, consider the problem of sorting n numbers from smallest to largest. The size of an instance would be n. So for an algorithm to run in polynomial time the running time must be bounded by a polynomial in n. To the reader who would like to know more: the Wikipedia page on time complexity is a good starting point.

In Algorithm 1 we have given the pseudo code of the generic local search method.

Algorithm 1. *Local search method.*

Input:
A neighborhood $\mathcal{N}(x)$ for each $x \in X$ where X is the state space,
a method $SELECT$ to select a neighbor from a neighborhood,
an acceptance criterion ACC depending on the current and previous state,
a stopping criterion $STOP$ depending on (potentially) all previous states and
an initial state x_0.

Initialization:
Use $SELECT$ on $\mathcal{N}(x_0)$ to select a neighbor x_0^*.
if *$ACC(x_0^*, x_0) == true$* ***then***
    Set $x_1 = x_0^*$
else
    Set $x_1 = x_0$
end if
Set $i = 1$
while *$STOP(x_0, x_1, \ldots, x_i) == false$* ***do***
    Use $SELECT$ on $\mathcal{N}(x_i)$ to select a neighbor x_i^*.
    if *$ACC(x_i^*, x_i) == true$* ***then***
        Set $x_{i+1} = x_i^*$
    else
        Set $x_{i+1} = x_i$
    end if
    i:= i+1;
end while

Output:
The best state x_i.

In the definition of a local search method we mention an acceptance criterion, it is defined as follows.

Definition 2.2.2 (Acceptance criterion)**.** An acceptance criterion is a rule used to decide if the move to the evaluated neighbor of the current state should be made.

The acceptance criterion greatly determines the performance of the local search method. In fact the acceptance criterion is what distinguishes two important classes of local search methods differ. We will describe these two classes and their acceptance criteria as examples below.

Example (Greedy search)**.** *A Greedy search method uses the most basic acceptance criterion: a neighbor is accepted if it has a lower cost than the current state.*

Example (Simulated annealing)**.** *This type of method uses a slightly more elaborate acceptance criterion. If a neighbor has lower cost than the current state*

it is always accepted, if it has a higher cost it is accepted with the following probability:

$$P(e, e') = \exp(-\frac{e' - e}{T_k}) \tag{2.2.2}$$

Where e is the cost of the current state, e' the cost of the evaluated neighbor and T_k a parameter known as the cooling temperature depending on the iteration number k. The name cooling temperature indicates the relation between the metallurgic process of annealing. The cooling temperature is assumed to decrease in the number of iterations. This makes the probability that a worse solution is accepted increasingly smaller. The reason to sometimes also accept if the neighbor is a worse solution is to escape local optima.

By now we have mentioned it a few times, a local search method moves from one state to another. The direction in which we move is determined by what the neighbors of the current state are. These neighbors are called a neighborhood.

Definition 2.2.3 (Neighborhood)**.** A neighborhood is a set of solutions which are related to the state by some predefined operator or by any of a few operators.

There is of course the question of what would make a neighborhood a good neighborhood? From the definition it seems that any selection of solutions can form a neighborhood, however not every selection should result in a good search method. Before answering this question we first give a few examples of neighborhoods to get more familiar to the concept.

Example (Neighborhood)**.** *A standard example for a neighborhood is one for the Travelling Salesman Problem (TSP). The TSP is the problem of finding a shortest route between a given set of cities such that each city is visited once and the last city is equal to the first city. A state s can be noted as a permutation of the cities. A possible neighbor of s could then be a state s' in which two of the cities have swapped places in the permutation.*

The following is a (first version of a) neighborhood used in the project.

Example (Neighborhood SmallCellRemoval)**.** *This neighborhood allows us to remove unnecessary sites from our network configuration. To determine which sites are unnecessary we first determine the average load per site. We do this by summing the load of all cells belonging to a site and dividing it by the amount of sectors (cells) the site has. We then sort the average loads per site and consider the active site with the smallest average load. If this load is less than* 0.35 *we adjust this site according to the following rules:*

- *If the site was a 6-sector site we reduce it to a 3-sector site.*
- *If the site was either a 3-sector site or a micro site we remove it completely.*

The above are all examples of neighborhoods derived by the use of one operator. In the last example we could extend the neighborhood by defining another operator which does more or less the opposite: it adds an upgrade to the network around a site with a load above the threshold.

Now back to the question, what would make a neighborhood a good neighborhood? Ideally it would contain the global optimum and nothing else, since

then we would only need to make one step before we are done. It is clear that this will never be the case if you decide to use a local search method. The goal in designing neighborhoods is however to approximate this situation as well as possible. By using the structure of the problem it is often possible to make sure that at least some part of the neighborhood is expected to have a better objective value than the current solution. Designing a good neighborhood or good neighborhoods also means that you do not want to restrict yourself to solutions that are sub-optimal. A good neighborhood structure would make the search space connected. In this context connected means that from any state it is possible to make a walk to any other state where each step moves to a neighbor.

This brings us to a bit of terminology. The following two definitions will be useful when discussing our local search methods and the results later on.

Definition 2.2.4 (iteration)**.** The proces of selecting a neighbor, randomly, of the current state and evaluating its performance. The iteration ends by either accepting or rejecting the neighbor which provides the state for the next iteration.

This definition is a very important one since it is not the only way to select the next state. An alternative way would be to evaluate all neighbors of the current state and then select the next state based on their objective values. We choose to not evaluate all neighbors since an evaluation is time expensive.

Definition 2.2.5 (run)**.** We call a run the application of the local search method to the problem. By this we mean the entire proces from initial solution to final solution. For now we end our local search method after a fixed number of iterations. Eventually we will also implement different stopping criteria (for example not finding an improvement for 50 iterations).

A final and important question is: When does a local search method terminate? Ideally it would only terminate when the optimal solution is found and also as soon as the optimal solution is found. This would mean that the algorithm needs to be able to recognize an optimal solution. In many cases where local search is used this is not the case. This means that we need to have some kind of criterion which can be used to determine if the algorithm should terminate.

Definition 2.2.6 (Stopping criterion)**.** A stopping criterion for the algorithm is a criterion evaluated in each iteration, if it is fulfilled the algorithm stops.

We will now present examples of types of stopping criteria.

Example (Stopping criteria)**.**

- *A predefined number of iterations is reached.*
- *A predefined computation time has been exceeded.*
- *Not finding an improvement for a certain number of iterations.*
- *Not finding a (relatively) large improvement for a certain number of iterations, i.e. compare the cost of the best found solution k iterations ago with the cost of the best found solution at the moment.*

- *The distance between consecutive improving solutions is small. This would require the definition of a meaningful metric on the solution space.*
- *If the optimal value is known the algorithm could terminate upon finding a solution with that value or relatively close to that value.*

This list of stopping criteria is not complete. Often a combination of different stopping criteria is used. Usually one of the first two criteria is used together with another one. The first two ensure that the algorithm does not go on forever.

In the next section we discuss in more detail the Greedy, Simulated Annealing and Tabu search classes. We have not yet seen Tabu search in an example, it will be introduced and explained in the next section.

2.3 Properties of various types of local search methods

In this section we describe various types of local search methods. For each method we will list its key characteristics and the advantages and disadvantages of the method. In the previous section we have used Greedy search and Simulated Annealing as examples of search methods, the third class we will discuss is that of Tabu search methods.

2.3.1 Greedy search

A Greedy search method uses the most basic acceptance criterion: a neighbor is accepted if it has a lower cost than the current state. There are no restrictions on the neighborhoods. There is no use of memory, that is: information about the previous states is not taken into account when selecting the next state. The main advantage of this class is that it is easy to implement, it also does not use any search history which simplifies the selection process. A well known disadvantage is its tendency to get stuck in local optima. This type of method can be compared to a person climbing a mountain. In this analogy a better state would be a higher position on the mountain. A move between states would be literally taking a step. The acceptance criterion means that a step can only be taken uphill. If the mountain is nice and smooth, with only one peak, this kind of walk would bring you to the peak, the global optimum. But if the mountain has multiple peaks, with different heights then it is clear that once a person reaches a peak it stops. The first peak visited is not necessarily the highest peak, which means we can end up in a sub-optimal state.

2.3.2 Simulated Annealing

Similar to Greedy search the typical Simulated Annealing method does not have any restrictions on the neighborhoods nor does it use any memory. This type of method uses a slightly more elaborate acceptance criterion. If a neighbor has lower cost than the current state it is always accepted, if it has a higher cost it is accepted with the following probability:

$$P(e, e') = \exp(-\frac{e' - e}{T_k}) \tag{2.3.1}$$

Where e is the cost of the current state, e' the cost of the evaluated neighbor and T_k a parameter known as the cooling temperature depending on the iteration number k. The cooling temperature is assumed to decrease in the number of iterations. This makes the probability that a worse solution is accepted increasingly smaller. The reason to sometimes also accept the neighbor if it is a worse solution is to escape local optima. Going back to the previous analogy of a hill climber this type of acceptance criterion would mean that the hill climber would sometimes move to a lower position, in the hope that later on it can move to a higher position than where it was. The longer the person is already climbing the less likely he is to move to a lower position. A further introduction of this method is given in [45].

The advantage of the method is already mentioned, it is possible to escape a local optimum. In [17] some important disadvantages are mentioned at the end of the (lengthy) article. The cooling temperature has to be chosen in such a way that it decreases sufficiently fast to allow acceptable running times but at the same time it has to provide enough freedom to allow the algorithm to escape local optima. The best choice depends on the problem type and size, which means it is very difficult to get exactly right without experimentation. Especially when there are only a limited number of iterations available the benefits from this method will be negligible compared to the Greedy search category. In [22] some convergence results are shown. They look at the process as a Markov chain and show that under certain conditions on the initial choice of the cooling temperature and the rate of decay of the cooling temperature there is asymptotic convergence to an optimal solution. This result does require some properties from the neighborhood structure, it requires the search space to be connected in the sense that every state can be reached from any state with a positive probability. For the readers more familiar to Markov chains this makes the Markov chain irreducible. Another article which looks at convergence results (with similar findings) is [27].

Finally we note that both Simulated Annealing and Greedy search belong to the broader class of threshold algorithms (also introduced in [22]). The general form of a threshold algorithm is the same as our description of a local search method with the following acceptance criterion:
When in state i we accept neighbor j if

$$f(i) - f(j) < t_k$$

where t_k is the threshold in iteration k.

We can see that this comes down to Greedy search when we take $t_k = 0$ for all k and to Simulated Annealing when we take

$$t_k = \begin{cases} \infty & \text{with probability } \exp(-\frac{e'-e}{T_k}) \\ 0 & \text{else} \end{cases}$$

2.3.3 Tabu search

In [24] Glover was the first to introduce Tabu search. In [25] the method of Tabu search was discussed with as an example the capacitated plant location problem, in a later chapter we will remark that the problem this thesis focuses on is closely

related to that problem. Tabu search is inherently different from the previous two classes since it does use knowledge about the previously visited states. The aim of this method is to avoid cycling between states. It does so by declaring certain moves forbidden, i.e. tabu, based on the previous steps. These forbidden moves are stored in a tabu list which is updated after each iteration. There is a lot of freedom in deciding which moves should be forbidden. The easiest example is to declare the previous state tabu. This can easily be extended by letting the tabu list contain the last m explored solutions. In the case that your neighborhood consists of several operators[2] you could also declare an operator tabu. This could for example be useful when you see that the application of one operator has been rejected multiple times in a row, you could declare it tabu and hope that another operator will provide a better solution, after that you could remove it again from the tabu list.

Tabu search is often used with the greedy acceptance criterion, but other acceptance criteria could be used as well. When this is the case often a combination between that method and the phrase 'with memory' is used, for example Simulated Annealing with memory.

An important observation is that in the case of the greedy acceptance criterion we could run into a problem with our local search. Suppose we reach the optimum in some iteration k. In every following iteration we will reject the move and subsequently our tabu list could potentially grow to contain the entire neighborhood of the optimum. In that case the algorithm should of course terminate. Since we do not know beforehand if we have reached the optimum it is common practice to restart the method from a random solution when this situation occurs (unless the stopping criterion specifies otherwise).

The intended advantage of Tabu search compared to Greedy search is the ability to learn from previous 'mistakes' (rejections) and thereby focus on the more promising areas of the search space. There lies a challenge in choosing the tabu list in such a way that the search does not become too restricted to a certain direction whilst learning as much as possible from previous 'mistakes'.

2.3.4 Overview

In the previous section we have given a short introduction to the methods Greedy search, Simulated Annealing and Tabu search. The first one is the most basic method, it is therefore often used to benchmark the others in articles where a comparison is made. Its disadvantage is that it might get stuck in local optima. Simulated Annealing attempts to avoid local optima by selecting a worse neighbor with a certain probability, where the probability decays with the number of iterations. There exist some asymptotic convergence results but in practice the method is mostly valuable when a large number of iterations is acceptable. The Tabu search method attempts to learn from the previous states and/or attempted neighbors by declaring certain moves forbidden. It should avoid cycling completely. This property does come at a cost: memory storage. In some applications this is not a problem, the required memory is negligible (in the application in this thesis this is the case).

The way to design a good neighborhood structure has not yet been discussed. Since the design of a neighborhood is very problem specific it is not possible to

[2]Previously also described as a neighborhood structure consisting of multiple neighborhoods, an operator refers to a neighborhood.

say something useful for general problems (other than what we have mentioned in the previous section), in Chapter 4 we will design a neighborhood structure for the problem this thesis deals with, described in Chapter 1. We do want to stress that the design of a good neighborhood is crucial to the performance of the methods described above.

A concluding remark about local search methods: the methods described above are sometimes called meta-heuristics since they are in fact a heuristic way to steer the heuristic process of searching locally.

2.4 Integer Linear Programming & relaxations

Linear programming has been proven to be a very important mathematical optimization technique. In Chapter 5 we use related techniques to find a good initial state for our local search method. In this section we introduce the reader to linear programming, (mixed) integer linear programming and binary programming. For each of the mathematical programs we will give its general formulation and, to illustrate, we treat a toy numerical case throughout this section.

2.4.1 Linear Programming

Linear Programming, LP, consists of mathematical programs for which the objective function is linear and the constraints are written as a linear system. In Equation (2.4.1) the general form of an LP is given where $c \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Note that the inequality $Ax \leq b$ is a vector inequality, for us this means that each coordinate has to satisfy its inequality.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{2.4.1}$$

The **feasible region** of a mathematical program is defined as the set of points which satisfy the constraints. In the general LP formulation this is equal to

$$\{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}.$$

The following is a very simple example of an LP problem.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & -x_1 + x_2 \\ \text{subject to} \quad & x_1 + 2x_2 \leq 3\frac{1}{2} \\ & x_1, x_2 \geq 0 \end{aligned} \tag{2.4.2}$$

This example uses two variables (x_1, x_2) and has three constraints ($x_1 + 2x_2 \leq 3\frac{1}{2}$ and $x_1, x_2 \geq 0$). As we have only used two variables we are able to draw the feasible region, see the blue region in Figure 2.1. That this is indeed our feasible region can be seen quite easily. First of all we have the constraints $x_1, x_2 \geq 0$ which indicate that our feasible region is contained in the non-negative quadrant of the (x_1, x_2)-plane. Secondly the constraint $x_1 + 2x_2 \leq 3\frac{1}{2}$ induces a boundary

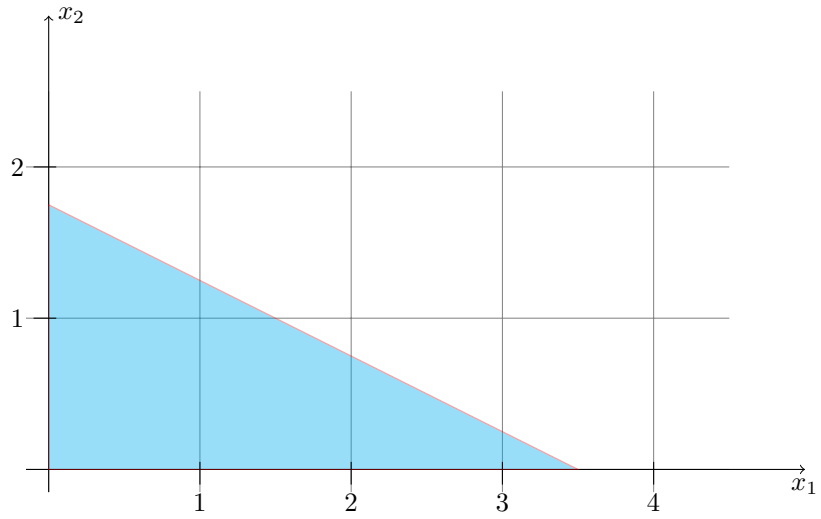

Figure 2.1: The feasible region of the system given in Equation (2.4.2).

line $x_1 + 2x_2 = 3\frac{1}{2}$. We can clearly see that this boundary line intersects the x_1-axis at $x_1 = 3\frac{1}{2}$ and the x_2-axis at $x_2 = 1\frac{3}{4}$. Since $(0, 0)$ also satisfies the constraint $x_1 + 2x_2 \leq 3\frac{1}{2}$ we clearly have the blue region in Figure 2.1 as our feasible region.

2.4.2 Integer Linear Programming

It can happen that the LP formulation presented in Equation (2.4.1) is inadequate for the problem you want to solve. Suppose for instance that the example of the previous paragraph, Equation (2.4.2) represents the problem of deciding how many apples (x_1) and oranges (x_2) to buy to minimize the amount of oranges minus the amount apples. Any non-integer point in the feasible region would represent a meaningless solution, assuming you cannot buy half an apple for instance. To model this problem correctly we would require x_1 and x_2 to be integer, i.e., we add the constraint $x_1, x_2 \in \mathbb{N}$. Adding such a constraint to our general LP formulation gives us an Integer Linear Program, ILP.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{N}^n \end{aligned} \tag{2.4.3}$$

Adding such a constraint to the example of the previous paragraph gives us the following ILP.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & -x_1 + x_2 \\ \text{subject to} \quad & x_1 + 2x_2 \leq 3\frac{1}{2} \\ & x_1, x_2 \geq 0 \\ & x_2, x_2 \in \mathbb{N} \end{aligned} \tag{2.4.4}$$

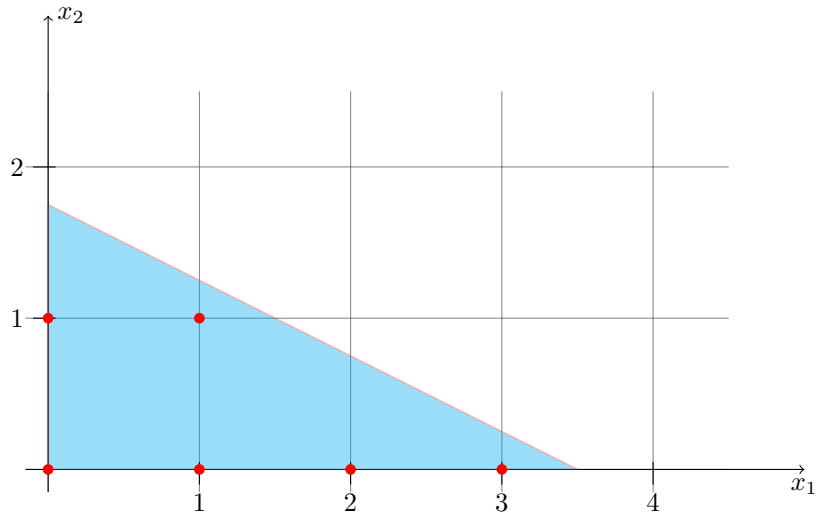

Figure 2.2: The red dots indicate the feasible region of the ILP in Equation (2.4.4), the shaded area is the feasible region of the corresponding LP problem.

The feasible region of this ILP can be seen in Figure 2.2. Note that the optimal solution of the ILP (2.4.4) is the point $x_1 = 3, x_2 = 0$ with cost -3 whilst the optimal solution of the same problem without the integrality constraint (see (2.4.2)) was $x_1 = 3\frac{1}{2}, x_2 = 0$ with cost $-3\frac{1}{2}$. It is not a coincidence that the ILP yields a higher cost than the corresponding LP obtained by relaxing the integrality constraints (for minimization!). The process of relaxing certain constraints is called **relaxation**. The goal of a relaxation is to make the problem easier to solve. For instance relaxing an ILP to an LP allows you to solve the LP with the simplex method (in practice a fast solution method), whereas the simplex method is not available for ILPs. Of course the obtained optimal solution of a relaxation is not always valid for the original problem, the relaxed constraint can be violated by the optimal solution of the relaxation (this is the case in the example above). In Chapter 5 we will use relaxations often, we will pay special attention to what it means for a potential solution to relax a constraint.

Remark (Binary programming). An integer linear program with the added constraint that all variables are bounded between zero and one is called a **Binary Program** or sometimes a $0-1$ integer linear program. Binary programming is an NP-hard problem, it was one of Karps 21 NP-complete problems [33]. Since integer linear programming is a generalization of binary programming it is also an NP-hard problem.

2.4.3 Mixed Integer Linear Programming

A generalization of both linear programming and integer linear programming is Mixed Integer Linear Programming, MILP. In a MILP we have two sets of variables, the first we allow to be continuous, as in an LP problem, the second we restrict to the integers, as in an ILP problem. The general formulation is

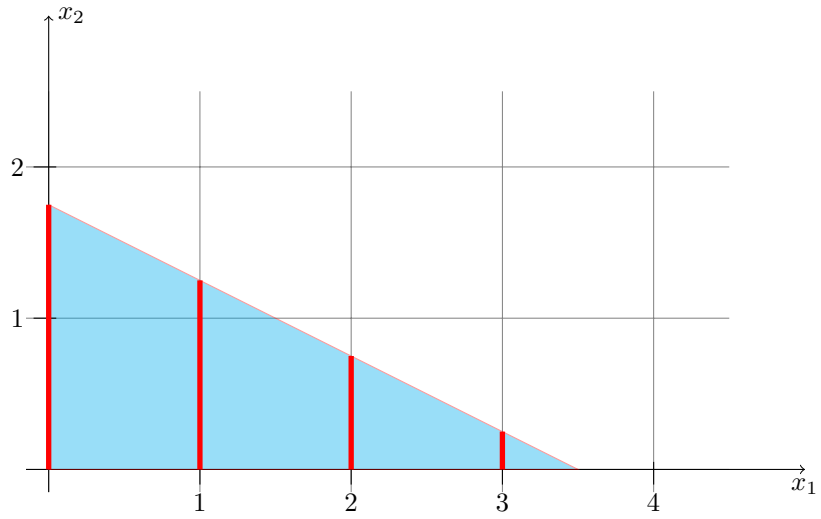

Figure 2.3: The red lines indicate the feasible region of the MILP in Equation (2.4.6), the shaded area is the feasible region of the corresponding LP problem.

given below.

$$\begin{aligned} \underset{x,y}{\text{minimize}} \quad & c_1^T x + c_2^T y \\ \text{subject to} \quad & A\begin{pmatrix} x \\ y \end{pmatrix} \leq b \\ & x, y \geq 0 \\ & y \in \mathbb{Z}^{n_2} \end{aligned} \tag{2.4.5}$$

where $c_1 \in \mathbb{R}^{n_1}, c_2 \in \mathbb{R}^{n_2}, A \in \mathbb{R}^{m \times (n_1+n_2)}$ and $b \in \mathbb{R}^m$.

In the example LP problem, Equation (2.4.2), we can for instance require x_1 to be integer. This yields the following problem:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & -x_1 + x_2 \\ \text{subject to} \quad & x_1 + 2x_2 \leq 3\frac{1}{2} \\ & x_1, x_2 \geq 0 \\ & x_2 \in \mathbb{N} \end{aligned} \tag{2.4.6}$$

with a feasible region as in Figure 2.3.

Chapter 3

Evaluation tools

The formulation of the problem in Section 1.3 clearly shows that a method to evaluate the performance of a network configuration is needed. In the remainder of this thesis we will use one evaluation tool. This tool is called SONlab and was developed by atesio [3].

In this chapter we give some background information on network evaluation tools. First of all we explain different classes of network evaluation tools: static and dynamic simulators and a static analysis. SONlab is of the third type. The remainder of this chapter discusses SONlab. In the preface to this thesis we have already mentioned it, during this thesis work another evaluation tool was developed. It is of the same type as SONlab: a static analysis tool. We have described this tool in Appendix D. We would advise the reader to read this chapter and whenever you see a model you can take a look at Appendix D and see how we did it.

The models used in SONlab can be divided in three categories:

1. The network aspects
2. The propagation environment
3. The traffic handling

This chapter will describe these three categories in this order.

3.1 Introduction to mobile cellular network evaluation methods

There are different ways to evaluate a mobile cellular network. Each has its own advantages and disadvantages. In this section we will highlight these.

The first way to look at a mobile cellular network is to take a snapshot of a situation and calculate the KPIs based on that snapshot. This method is called a **static simulation**. Calculating the KPIs based on one snapshot means that you can be very unlucky if for example the snapshot contains a lot more users than there are active on average. A way to avoid this, and the second way to look at a network, is to consider a larger time frame. The larger time frame can be measured by taking multiple snapshots at different times, a small time step

apart. The resulting evaluation is called a **dynamic simulation**. A dynamic simulation can also take into account mobility of users because the different snapshots are correlated. It is trivial that a dynamic simulation of a network requires more computation time than a static one. These two methods are called simulators because they take snapshots of a simulation of the network. The third way to look at a mobile cellular network is called a **static analysis**. A static analysis does not look at individual users but at area based statistics. The area based statistic that we use is the average call length, or in other words: the average traffic demand. As the name suggests we are interested in the average traffic demand in a certain area. That means that we divide the total area under consideration in small regions. In the remainder of my thesis we will call these regions pixels or service test points. Each pixel is has a certain average traffic intensity. The computations that we do based on these pixels are very similar to what is done in a static simulation, if we see the pixels as the active users in the snapshot. In Appendix A we discuss these computations in more detail. The load that we compute in this way is not the actual load at any moment in time but rather an average. From now on we will call this average load again load for simplicity. Taking the average load over a certain period of time means that we lose some data about the peak loads. If the average load is, for example, 0.7 then there can be moments where the actual load approaches one. A situation where the load approaches one is highly undesirable since this would cause serious problems for users in that area (calls being blocked for example). In our optimization approach we take this into account by using a threshold which must not be exceeded that is lower than one. In the remainder of this thesis a threshold of 0.6 is used, but this is actually an input that should be provided by a network operator.

3.2 Network aspects

The mobile cellular network is formed by two sides: a demand an a supply.

The demand is formed by the users in a network. As we have mentioned earlier SONlab looks at area based statistics so the demand is formed by **a pixel map of the traffic distribution**. It is unclear which units the traffic distribution in SONlab has. In [18] the following explanation is given (on page 52). *The traffic intensity map in SONlab should be seen as a relative intensity map. A linear scaling factor is applied to get to an offered traffic map in which each pixel has a certain level of offered traffic in bps. The scaling factor is a parameter we can change before starting our simulations.* The problem is that only one scaling factor would actually correspond to bits per second. In Section 5.4 we discuss this problem in more detail and there we explain how we dealt with it. If the input is truly in bits per second then the amount of traffic a base station can handle is way too low. An input in terms of Mbps would be more realistic.

The supply side is formed by the base stations. We refer the reader to Appendix A if any of the following concepts are unclear. Each base station consists of several antenna's aimed in a different direction. Each antenna has a set of parameters:

- A location. That is, (x, y, z)-coordinates. Where the z coordinate is the height above ground level of the antenna.

- An azimuth.
- A tilt.
- A power setting.
- An antenna type.

The importance of the antenna type will become clear in the next section when we describe the antenna gain. The scenarios we have used in SONlab use one antenna type per type of base station and three types of base stations: 3-sector macro, 6-sector macro and micro.

Another aspect of the network is the performance. This is captured in the Key Performance Indicators (KPIs), that we have described in Section 1.1. SONlab is able to provide us with the load per cell. Further output that can be obtained is:

- A pixel map with the received Signal to Interference plus Noise Ratio (SINR)
- A pixel map with the maximum data rate a user could receive
- Per pixel a list of the strength and cell ID of the strongest received reference signals
- Per pixel a list of the assignment probabilities of the pixel to the cells mentioned above

3.3 Propagation environment

Radio waves propagate through a medium and therefore there is a certain loss in signal strength between the transmitted signal at the base station and the received signal by the user. This loss can be seen as the sum of several losses (when signal strength is considered in decibel). The first and largest loss is due to distance and terrain influences (buildings, trees, etc.), for this we use path loss models. These path loss models give an average loss. However there is also variation from this average. This is modeled with shadowing. The last loss is confusingly called the antenna gain, it reflects the radiation pattern of an antenna. In this section we discuss these three types of losses in this order.

3.3.1 Path loss

A path loss model is used in SONlab to get an average loss of signal strength between transmission at the base station and reception by the user. However, we use pixels instead of individual users. The path loss we obtain is thus an average path loss over the entire pixel. The path loss first of all depends on the distance between the base station and user. This relation is a decay in distance of the form d^{α} where d is the distance and α a constant usually between 3 and 4. The path loss is also influenced by the terrain between the base station and user. For illustration, in Figure 3.1 we have given a sketch of the propagation paths that are possible between a mobile user and antenna.

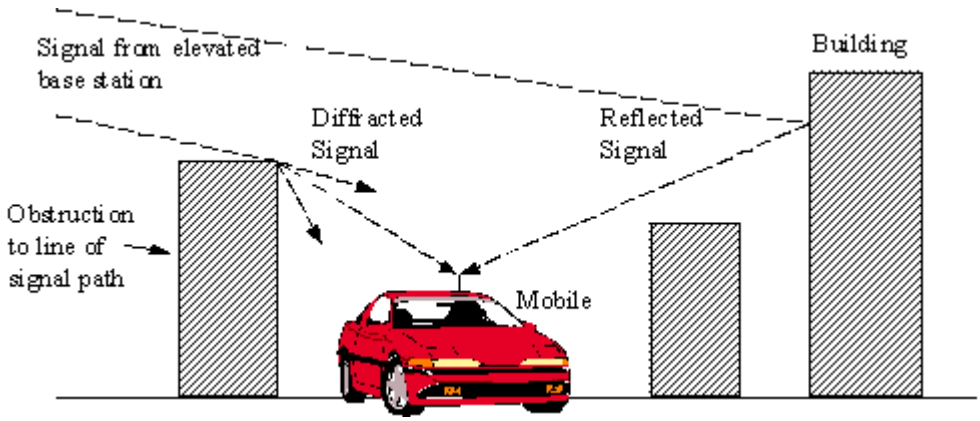

Figure 3.1: Radio propagation.

SONlab uses detailed information about the terrain between a base station and user. In fact, this is the main reason why SONlab provides such realistic analysis. In Section 2.3 of [18] the path loss models used in SONlab are described. The path loss is calculated using a ray tracing model[1]. The ray tracing model is divided in two sub-models:

- A vertical plane model that accounts for losses in the vertical plane between base station and user
- A multi-path model that accounts for losses experienced due to signals reaching the user from multiple directions not exactly at the same time.

The vertical plane model distinguishes between two cases: there is a direct line of sight from antenna to user or not. See also Figure 3.1. If there is a direct line of sight, then for small distances free space propagation loss is used (a decay cubic in the distance) and for larger distances a version of the Okumura-Hata model is used. In Appendix D we discuss the simulator developed during this thesis work. It uses the Okumura-Hata model as well and therefore we would like to refer to that appendix for a description of the Okumura-Hata model. If there is no direct line of sight there is a diffraction loss for propagation over rooftops.

The multi-path model represents losses in the horizontal plane. It uses even more detailed information about terrain since reflections and scattered paths off of buildings are considered. Reflected paths are considered up to the second reflection, with a maximum image source distance of 1 km. Scattered paths are only considered when the transmitter and receiver are within 500 m of each other. This multi-path model can only be applied when detailed 3D information is available, within the SEMAFOUR project such data is available for a certain region in Hannover.

If the reader is interested in more information about the models used in SONlab we would like to refer to Section 2.3 of [18] and the references therein.

[1] We are only interested in the outdoor path loss, since in our scenario all users are outdoors (see Chapter 6 for a description of the scenario). In [18] there is also some reference to an outdoor-to-indoor loss.

3.3.2 Shadowing

In the previous subsection we have described the path loss model used by SONlab. This path loss model is an approximation of the real path loss in several ways. There is an error in the way obstacles are taken into account and the path loss is given per pixel but there can be variability within a pixel. Shadowing is used to model the variance of the path loss that occurs due to these errors.

Roughly speaking shadowing can be divided into two classes: fast and slow shadowing. The latter is also called long-term shadowing. Fast shadowing, or multi-path fading, is due to moving objects and thus acts on a small timescale. We use area based statistics in our user model, that means we average over time. The fast shadowing is thus averaged out (the mean is assumed to be 0). Slow shadowing is the one we are interested in. From now on we simply say shadowing when we mean slow shadowing. Shadowing is modeled as a stochastic variable which can be added to the calculation of the received field strength. For each point on our grid we have, for each BS, a shadowing term.

If we consider one user (or test point) and one antenna we have one shadowing term S'. It is widely accepted (and supported by measurement studies) that S' follows a log-normal distribution when represented in linear units. In [44] a theoretical basis is given for choosing the log-normal distribution. This means that if we consider the field strength in dB, and call the corresponding shadowing S, we have that S follows a normal distribution. So $S \sim N(0, \sigma^2)$ where typically $\sigma \approx 6$ dB for urban environments.

One user, however, receives the pilot signal from many base stations, say base station 1 up to k. For each base station we then have a shadowing term: $S_i \sim N(0, \sigma)$ for $i = 1, \ldots, k$. It is then interesting to look at the joint distribution of $S = (S_1, \ldots, S_k)$. S is again normally distributed, but then by a multivariate normal distribution. That is

$$S \sim N_k(0, \Sigma)$$

where Σ is a matrix giving the covariance between the variables. We have

$$\Sigma = \sigma(S, S) = E[(S - E[S])(S - E[S])^T] = E[SS^T] - E[S]E[S]^T.$$

Note that on the diagonal this just reduces to the variance of each of the S_i.

The S_i all capture a correction upon the signal loss experienced over the area between the base station and mobile user. These areas are not disjoint, they overlap in the region close to the mobile user. The S_i are therefore correlated, this type of correlation is called cross-correlation. In Figure 3.2 we have sketched the situation with two antennas. Here ϕ is the angle between the two line of sight paths, d_1 and d_2 are the respective distances from antenna 1 and 2 to the mobile user. The smaller the angle, the greater the overlap and therefore we would like to see a high correlation for small angles. On the other hand there is also the distance to consider, if both antennas are located at the same distance from the mobile user we expect the correlation to be largest. This is because when the distances are the same the fraction of the area close to the user over the entire area between the base station and user is the same. Remark that for two sectors belonging to the same base station the correlation has to be one (since the above mentioned areas are the same).

In the previous paragraph we have indicated that there needs to be correlation between the shadowing terms of different base stations for one user (or

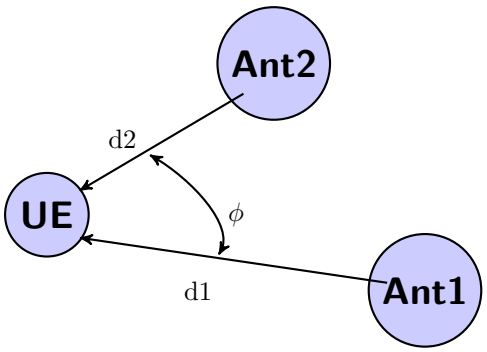

Figure 3.2: A situation with two antenna (1 and 2) and a mobile user (UE).

test point). However, since the test points themselves are related as well (they cover a geographic region), we would also expect correlation between test points which are close to each other. In the literature this is called auto-correlation . The term 'auto' is a little bit confusing since we are talking about two different test points. The origin of the term auto-correlation lies in the following point of view. Two different test points can be viewed at as being the same user, moving at a certain speed, at different times.

This is unfortunately all we can say about the shadowing in SONlab. We know that SONlab uses a log-normal shadowing term however we do not have access to any specific information about the distribution or the correlations we have mentioned above (we even do not know if they assume correlation). In Appendix D we show how shadowing can be modeled and is modeled in the evaluation tool we developed.

3.3.3 Antenna gain

The last loss we need to discuss is confusingly called a gain: the antenna gain. The antenna gain accounts for the directivity of the antennas. Antennas are aimed in a certain direction and users are not always exactly in that direction. The line between user and antenna has a certain offset (angle) in the horizontal and vertical direction compared to the direction the antenna is aimed at. In Figure 3.3 we have shown an example radiation pattern. The last two graphs show that the antenna gain can be divided in a horizontal gain and a vertical gain. The horizontal gain is the gain experienced due to the offset in the azimuth direction, the vertical gain is due to the offset in the tilt direction.

SONlab does not use all these details about the radiation pattern. It uses approximation formulas. One of them, the horizontal gain, is presented in [26]. The horizontal gain is also used in the evaluation tool described in Appendix D. The horizontal gain $G_h(\phi)$ [dB] depends on the angle in the horizontal plane between the center of the beam and the line between base station and user (in degrees). It is given by:

$$G_h(\phi) = G_m - \min(12\left(\frac{\phi}{HPBW_h}\right)^2, FBR_h)$$

where G_m [dB] is the maximum gain, $HPBW_h$[deg] is the horizontal half power

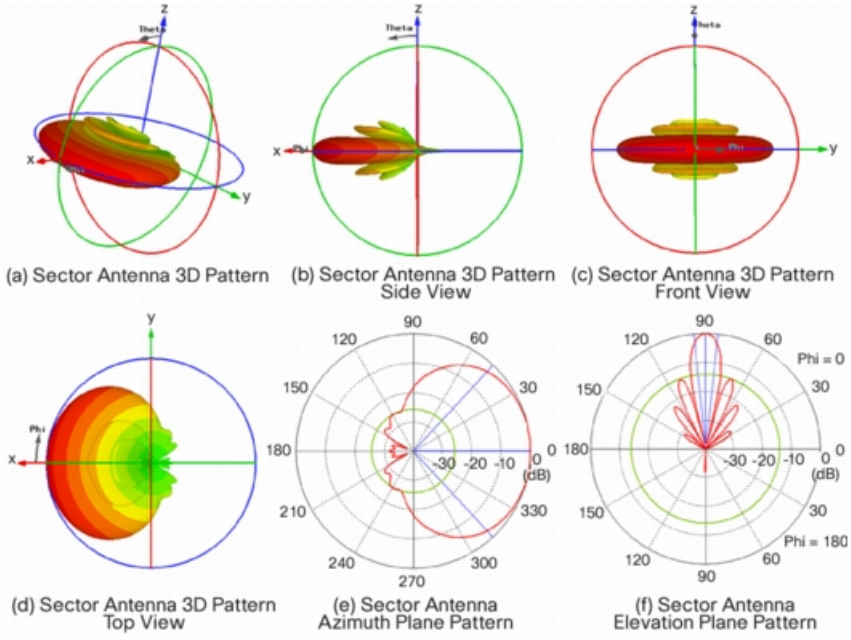

Figure 3.3: Radiation pattern of a 90 degree sector antenna. Source: Cisco.com, [4].

beam width and FBR_h[dB] is the Front Back Ratio[2] in the horizontal direction.

For the vertical gain SONlab uses a model different from the one described in [26]. The model SONlab uses is described in [18] and is as follows:

$$G_v(\theta) = \max\left(a(\theta, \theta_e), SLL_v\right)$$

where SLL_v [dB] is the Side Lobe Level and

$$a(\theta, \theta_e) = 20 \cdot {}^{10}\log(C(\theta) \cdot B(\theta, \theta_e))$$

with $C(\theta) = \sin(\theta)^3$ and

$$B(\theta, \theta_e) = \frac{1}{N_d} \frac{\sin(\frac{N\psi}{2})}{\sin(\frac{\psi}{2})}$$

with $\psi = \frac{2\pi d}{\lambda}\left(cos(\theta) - cos(\theta_e)\right)$. In these formulas θ is the analog to ϕ in the vertical plane and θ_e is the electrical tilt of the antenna (set to zero in our simulations!).

Remark. As we have mentioned before, SONlab uses a path loss model that takes into account multi-path propagation. The antenna gain formula presented in this subsection has to be applied to each of these propagation paths!

In Figure 3.4 we make a link between the approximation formulas and the radiation patterns we see in Figure 3.3. It is clear that the proposed formulas do not capture the small variations but the general picture remains intact. The figure shows the horizontal gain.

[2]The Front Back Ratio and Side Lobe Level will not be discussed in detail, from the formulas it should be intuitively clear what they mean.

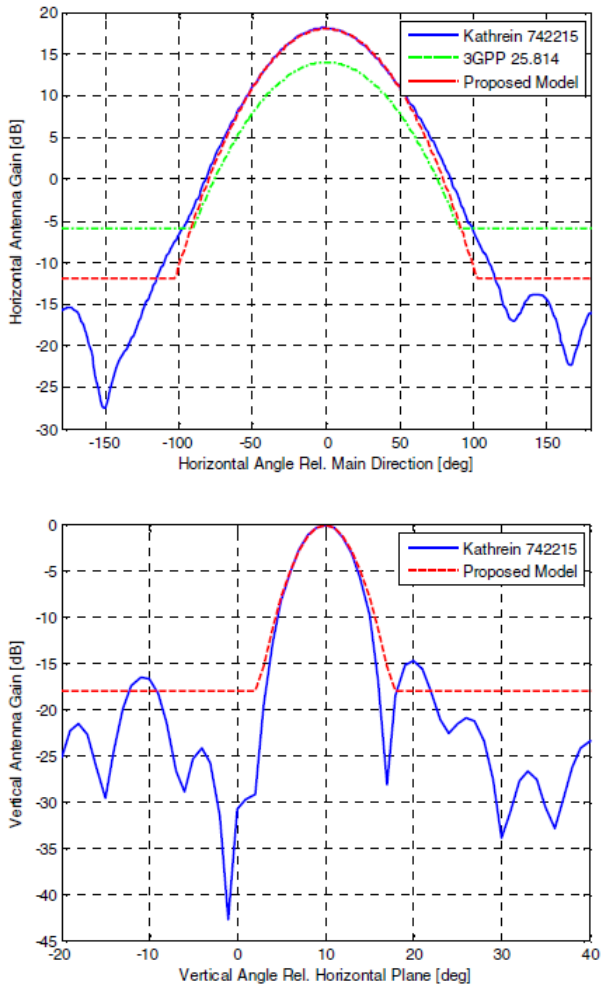

Figure 3.4: Comparison of the approximation formulas for the horizontal antenna gain to the radiation pattern of the Kathrein 742215 model. Source: [26].

3.4 Traffic handling

The assignment of users, or User Equipment (UE), to cells (cell assignment) is based on the best server principle. The best server of a user is determined as the cell for which the pilot signal reaching the user is the strongest. The pilot signal, also called a reference signal, is a signal that all cells broadcast at a specific frequency and a certain power setting. In the previous section we have thoroughly explained the losses that the reference signal experiences between transmission and reception. In particular we have seen that there is a stochastic aspect in our modeling: shadowing. SONlab therefore does not determine the best server deterministically, instead it provides an assignment probability of a pixel to a cell.

Once the cell assignment is completed the KPIs need to be calculated. In

this thesis we are interested in the load. In Appendix A.2 we have explained the load vector. There, we arrive at a fixed point problem. Atesio, the creators of SONlab, provided us with a link to the article *Analysis of Cell Load Coupling for LTE Network Planning and Optimization* [31].

In [31] the problem of determining the load of each cell in a network configuration was considered specifically for the LTE technology. They arrived at the same fixed point problem as we did in Appendix A.2. Let us denote this system by

$$\rho = f(\rho) \tag{3.4.1}$$

where ρ is the load vector. The function f maps the demand of each pixel in a cell to the load of that cell, based on the load of all other cells. In [31] they find the solution to this non-linear system by solving the following minimization problem:

$$\begin{aligned} \min \quad & \mathbf{1}^T \rho \\ & s.t. \rho \geq f(\rho) \\ & \rho \in \mathbb{R}^n_+ \end{aligned}$$

This problem is shown to be a convex optimization problem. It is shown that the optimal solution (if it exists) is indeed a fixed point of the mapping f and is even unique. In [30] a more thorough explanation of the model used in [31] is provided.

The fixed point problem is formulated using a deterministic cell assignment. However, SONlab uses assignment probabilities. These assignment probabilities can be used to create a fractional assignment of pixels to cells. Based on this fractional assignment the fixed point problem can be solved (in the function f the contribution of each pixel to each base station can be multiplied with this fraction).

3.5 Concluding remarks

The evaluations in the remainder of this thesis are performed with SONlab. As we have mentioned earlier the computation times SONlab requires are not too large. In Appendix B, Table B.1 an overview is given of typical computation times of various SONlab functions. The total computation times we see are roughly 40 seconds (except for the first step in which all cells had to be reconfigured). The functions we need from SONlab are in each iteration the Retrieve load function and in the first iteration of each instance the Get traffic grid function. This means that the total computation time in SONlab is usually less than 10 seconds per iteration (excluding the post-processing of the output data, i.e., the selection of a neighboring state).

Chapter 4

Local search approach

The problem defined in Section 1.3 has been shown to be NP-hard[1]. Therefore we consider heuristics to solve the problem[2], in particular we look at local search methods. Let us briefly recall our objective and state space.

Our goal is to find a network configuration for which the evaluation of a predicted traffic intensity pattern yields an acceptable performance (several KPI have to meet a minimum standard) against minimum financial cost. Our state space consists of a list of candidate sites with possible upgrades at each site. The existing network is a part of this list, but there are also some potential new sites. The possible upgrades we consider are the following. On new sites we can build a micro site and existing 3-sector sites can be upgraded to 6-sector sites (a process called 6-sectorization).

In this chapter we present our local search methods and explain how they can be generalized to use a concept called problem zones. We will present our local search methods by defining their key characteristics (cost function, neighborhood structure, acceptance criterion and stopping criterion, in that order). The presentation will be based on the basic greedy search method. The acceptance criterion and neighborhood structure have been varied and at the end of each of these sections the variations are discussed.

4.1 Cost function

Like any optimization method, the local search method needs a cost function to be able to compare different states. In our case, the cost function consists of two parts: one part measures the performance of the network and one part gives a measure to the financial cost of the selected network. The financial costs are based on the costs of the individual upgrade options, assumed to be given as operator input. Performance measures depend on the KPIs which triggered the DSS. If there are multiple KPIs (e.g. load, throughput), we enter the realm of multiobjective optimisation. To start we would combine the different KPIs into a single objective function by choosing a weighting factor for the different

[1]See Section 2.1 for an explanation of NP-hardness and Section 1.3.1 for references to articles proving the NP-hardness.

[2]Recall that we use solve in the context of heuristics even though heuristics do not always find a global optimum.

KPIs and adding them. Overall, the cost function will be constructed such that we search for upgrade options that satisfy the KPI targets against minimum financial costs.

As mentioned before, the cost function comprises of both financial costs and costs in terms of system performance or QoS. One part of our performance measure is the maximum cell load over all cells. Let us denote this maximum as Max_load. We want to find a solution where this maximum load lies below a predefined threshold (set to 0.6 in our scenarios). The following cost achieves this goal:

$$\max(\text{Max\_load} - 0.6, 0).$$

Another performance measure is based on the second KPI discussed in Appendix A.2, the overload traffic.

The overload traffic metric assigns to each pixel a fraction of its traffic intensity demand which is 'too much' for a cell with a load above the threshold. The second performance measure we have considered is the sum over all pixels of the overload traffic metric. The sum over all pixels reflects the performance problem in a nice way. If there are a lot of pixels with a certain overload, the performance of the network should be valued as worse than when there are less pixels with that overload. Also if one pixels has a certain overload the performance of the network should be worse when that pixel has a higher overload (if we keep all other pixels fixed).

The financial costs of a network configuration can then be modeled as the sum over the various site types of the weight of the type times the number of active sites of that type. That means that we count the number of active 3-sector, 6-sector and micro sites and add them with the given weights. Of course each network operator could use its own weights.

Nokia has provided us with reasonable weights[3], presented in the PhD thesis of Claudio Coletti [14]. The cost can be divided into three categories, the Capital Expenditures (CAPEX), the Implementation Expenditures (IMPEX) and the Operational Expenditures (OPEX). CAPEX denotes the investment costs (in terms of equipment), money which has to be spent only once. IMPEX relates to the costs made installing the equipment (site acquisition, deployment), this is also money which has to be spent only once. OPEX on the other hand is about returning costs, it envelops, amongst others, site rental, maintenance and electricity costs. In Table 4.1 the CAPEX, IMPEX and OPEX (one year) of several network upgrade options can be found (with the data from Fig. C.1 in [14]).

The costs presented in Table 4.1 can be combined into what is called the Total Cost of Ownership (TCO). As in [14] we take

$$\text{TCO} = \text{CAPEX} + \text{IMPEX} + 4 \cdot \text{OPEX}.$$

From the formula it is clear that we take the costs over a four year period (as the OPEX is multiplied by four). The TCO is then scaled to a relative TCO in such a way that the cheapest option has weight one. In Table 4.2 we present the TCO and relative TCO. In the remainder of this thesis we will use the relative TCO as weights to calculate the financial cost of a network.

[3]The cost estimates given are purely used for academic purposes. Even though Nokia has provided us with this information it does not mean that these numbers are also used by Nokia

Type of upgrade	CAPEX (in 10^3 €)	IMPEX (in 10^3 €)	OPEX (in 10^3 €)
Newly deployed LTE site	46	53,5	19,82
LTE site (reuse HSPA site)	35	9	10,92
LTE upgrade to 2nd carrier	26,5	5,75	2,31
Micro (High Cost)	8,5	6,45	7,79
Micro (Low Cost)	7,5	2,45	2,03

Table 4.1: CAPEX, IMPEX and OPEX in thousands of Euros for several network upgrade options.

Type of upgrade	TCO (in 10^3 €)	relative TCO
Newly deployed LTE site	178,78	9,9
LTE site (reuse HSPA site)	87,68	4,9
LTE upgrade to 2nd carrier	41,49	2,3
Micro (High Cost)	46,11	2,6
Micro (Low Cost)	18,07	1

Table 4.2: TCO in thousands of Euros and the relative TCO of several network upgrade options.

Finally, the financial costs and performance costs are integrated into one cost function. Remember we have chosen to formulate our problem as a minimization problem, our cost function should reflect this. We want to force the local search method to provide us with a solution which meets the KPI requirements, therefore any state which does not meet the requirements should have a cost higher than that of any state which does meet the requirements. We have chosen to do this in the following way[4].

- If the state meets the KPI requirements, then only the financial costs play a role. Based on the total number of upgrades available an upper bound on the financial costs is known. For ease of implementation we have scaled our financial costs such that the upper bound is much less than 10. This allows us to use 10 as an upper bound on the financial costs in each of our instances. This means that we do not have to change the upper bound each time we consider a new instance.

- If the state does not meet the KPI requirements, then we look at the performance measure. To ensure that the costs in this case are higher than that of the previous case we add a constant 10. The cost in this case will be:

$$\text{Cost\_performance(X)} = \sum_{\text{pixels } i} \text{Overload\_Traffic}_i \cdot I(\text{Overload\_Traffic}_i > O^*) \tag{4.1.1}$$

where I is the indicator function, O^* is a threshold on the overload, and $\text{Overload\_Traffic}_i$ is the overload traffic measure as described in Ap-

[4]Note that this is not the only way that we can ensure that feasible states have a lower cost than infeasible states.

pendix A.2.3:

$$\text{Overload\_Traffic}_i = D(x_i) \cdot \sum_{cells\, j : \rho_j \geq \rho^*} p(x_i, j) \cdot \frac{\rho_j - 0.6}{\rho_j} \tag{4.1.2}$$

where x_i is pixel i, $D(x_i)$ is the demand of pixel i, ρ_j is the load of cell j, ρ^* is the threshold load and $p(x_i, j)$ is the assignment probability of pixel i to cell j.

In formulas this gives us the following cost function:

$$\text{Cost(X)} = \begin{cases} C \cdot \sum_{i \in \{\text{site types}\}} w_i \cdot (\#\text{sites of type i}) & \text{if X satisfies KPI requirements} \\ 10 + \text{Cost\_performance(X)} & \text{else.} \end{cases} \tag{4.1.3}$$

Here C is the constant we use to scale our financial costs such that the upper bound is much less than 10. In our scenarios we take $C = 1/100$.

An important aspect of the search algorithm is the evaluation of the cost function in order to assess potential network upgrades. We use the SONlab simulation tool to determine the performance components of the cost function for a given state (network configuration) under the expected (future) traffic demand. SONlab provides a good trade-off between the speed at which the performance metric can be evaluated and the accuracy of the results. For more information about SONlab and simulators in general we refer to Chapter 3.

4.2 Neighborhood structure

The second and perhaps most important aspect of a local search method is the definition of the neighborhood structure. We have defined three different operators who together define the neighborhood structure. The neighborhood of a state will comprise of all states which can be obtained by applying one of the operators to the current state. Based on the current state we have to choose one of its neighbors to evaluate in the next iteration. We can not do an exhaustive search of the neighborhood due to time constraints so instead we will pick one according to a certain probability distribution. There first will be a certain probability to select each of the operators and then the operators themselves favor certain neighboring states based on the evaluation of the current state. In the remainder of this section we describe the three operators along with the distributions they use to select a neighboring state, finally we also describe the way to select an operator. The first two operators we describe are inspired by Hurley [29] and adapted to our state space.

The three operators are named and summarized as follows:

- *Traffic filler*; designed to improve the capacity of the network,
- *Small cell removal*; designed to reduce the financial costs by undoing unnecessary previously selected upgrades,
- *Swap*; designed to add some flexibility to the method in the situation where the current state meets the KPI requirements. The goal is to reduce financial costs by removing an expensive upgrade and replacing it by a cheaper upgrade.

Let us make one important remark before we continue. All of our local search methods draw without replacement from the neighborhood. That means that we keep a list of upgrades previously attempted and rejected, the upgrades from this list are ignored in later iterations (until a state is accepted, then we clear the lists). They are simply removed from the respective lists we mention in the description of the operators. This already comes close to a Tabu search method (see Section 2.3.3), but as it is the most logical thing to do we still call this our Greedy search method. Our Tabu search method will declare certain operators as forbidden.

4.2.1 Traffic filler

The Traffic filler operator is designed to improve the capacity of the network. As described in the introduction of this chapter we have two options available to reach this goal. We can perform 6-sectorization on 3-sector sites and we can install new micro sites. The question is which option do we consider and for which site? The input to this operator consists of the lists of sites currently active (one for each type) and of a location related to the problem area. This location depends on the problem characteristics, in particular the overload traffic measure. We then do the following:

1. The cell with the maximum load is identified. If its load is below the threshold, a failure of this neighborhood is reported and the operator terminates. This means the problem with respect to load is resolved.

2. If the maximum load is above the threshold and that cell belongs to a 3-sector site, 6-sectorization on that site is performed. If it was already a 6-sector site, a different site has to be altered.

3. Select a type of upgrade to perform. The distribution used is a probability of 0.6 to select 6-sectorization and a probability of 0.4 to select the activation of a micro site.

4. Based on the selected upgrade we compile a list of sites of that type. If the selected action is the activation of a micro site we consider micro sites that are no further than 1000m away from the problem location. Similarly, if we had elected to do 6-sectorization, we ignore 3-sector sites further away than 1500m of the problem location. These ranges are based on an optimistic estimate of the maximum range of a macro/micro site and the distance to the problem area. Based on other characteristics of the problem area obtained by the overload traffic measure the distances mentioned could be altered. This will be discussed in Chapter 6.

5. The selection of the specific site is based on the list obtained in the previous step and is done according to the following probability distribution: site j from the list is selected with probability

$$P(j) = C \cdot \mathrm{e}^{-2 \cdot d_j}$$

where d_j [km] is the distance of site j to the problem area and C is a constant used to make the sum of the probabilities over the list equal to one. The probabilities are chosen in a way as to give a higher probability

to sites closer to the problem location. Being close to the problem location will most likely mean a larger impact on the problem.

4.2.2 Small cell removal

The Small cell removal operator is designed to reduce the financial costs by undoing unnecessary previously selected upgrades. The question of which previously selected upgrade was unnecessary is difficult to answer. We can only determine if it was unnecessary by undoing it and evaluating again. The task at hand is thus to select a site for which the upgrade proves to be unnecessary.

1. The selection is done based on the KPI load. The load is computed at a cell level and the upgrades are done on a site level. Our first step is therefore to define the site_load, a metric on a site level, as the average load of the cells belonging to a site.

2. From the list of active sites we then select one according to the following probability distribution: site j is selected with probability

$$P(j) = C \cdot \mathrm{e}^{-2 \cdot \mathrm{site\_load}_j}$$

where $\mathrm{site\_load}_j$ is the site_load of site j and C is a constant used to make the sum of the probabilities over the list equal to one. The distribution is chosen such that sites with a lower site_load have a higher probability to be selected.

3. The previously performed upgrade of the selected site is undone. That is a 6-sector site will become a 3-sector site again and a micro site will be deactivated.

4.2.3 Swap

The Swap operator is designed to add some flexibility to the method in the situation where the current state meets the KPI requirements. The goal is to reduce financial costs by removing an expensive upgrade and replacing it by a cheaper upgrade. In principle this could also be done by first removing the expensive upgrade (for example by the Small cell removal neighborhood) and then in the next iteration perform a less expensive upgrade. The problem is that the removal of the expensive upgrade without the immediate replacement could lead to a state which does not meet the acceptance criterion (for example because the KPI requirements are met with the current state but not with the neighbor). Performing both steps in one iteration, as this operator does, solves that problem. To add some extra flexibility the operator is allowed to swap two upgrades of the same type (i.e. cost).

The formulation of this neighborhood will be more generic than those of the first two operators because in this case being more generic is easy and helps to understand what happens.

1. First the costs of all upgrade options need to be known. With $C(i)$ we denote the cost of upgrade option i. The probability with which we choose to remove an upgrade of option i is equal to

$$P(i) = \frac{C(i)}{\sum_i C(i)}.$$

Note that this probability is chosen such that options with a high cost have a high probability of being chosen.

2. Once the type of upgrade to remove is chosen we select a type of upgrade to do. The probability distribution we choose is the following. We sort the upgrade options with a lower cost than the selected option in the first step, from high to low. The most expensive option gets a weight of one, the second most expensive a weight of two, the third most a weight of three and so on. The probability with which we choose each option is then its weight divided by the sum of all weights. Remark that we can only select a type of upgrade to perform if its cost is lower than that of the type to be removed (no strict inequality, the same type can also be selected).

3. The choice of site to remove is then done in a similar way to the Small cell removal operator. From the list of active sites we select one according to the following probability distribution: site j is selected with probability

$$P(j) = C \cdot \mathrm{e}^{2 \cdot \mathrm{site\_load}_j}$$

where $\mathrm{site\_load}_j$ is the site_load of site j and C is a constant used to make the sum of the probabilities over the list equal to one. Note the difference lies in the positive exponent instead of the negative one in Small cell removal, this is because for this neighborhood it makes more sense to adjust important sites (those with a low load can probably be removed using Small cell removal).

4. The choice of site to upgrade is done as in the Traffic filler operator with the type of upgrade already determined, as location we take the coordinates of the removed site in the previous step. These coordinates are chosen such that the replacement has a good chance to keep the KPIs at the same level. In the situation where the type of upgrade to be performed is the same as the one to be removed, we, of course, do not allow an upgrade to be performed on the same site as where we removed an upgrade.

4.2.4 How to select the operator

By now it should be clear that the selection of an operator greatly determines which neighbors can be selected. We do something similar as what was done in [29], we select the operator based on the evaluation of the current state. We always attempt the Traffic filler operator. If it fails we know that the current state meets the KPI requirements (this was already known, but since it is also the first criterion checked by the operator it can just as well be tested by the operator). If it is successful we select that neighbor. Traffic filler is the only operator capable of providing an improvement in capacity (assuming Swap can not, because cheaper options are less effective) and as such it is logical to attempt this operator first. Next we have a distinction between two of our local search methods. The first one, our basic greedy method, selects Small cell removal with a probability of 0.75 and Swap with a probability of 0.25. Our second method, a 2-step approach, only uses the Traffic filler and Small cell removal operators, so

it always chooses Small cell removal after the Traffic filler has been attempted. The 2-step approach has not yet been introduced, it will be introduced in the next section about acceptance criterion.

Our basic greedy method has a probability distribution with which it chooses the operator. This probability was briefly investigated. We tested the greedy method on one scenario, the basic traffic scenario, with three different distributions ($25-75\%$, $50-50\%$ and $75-25\%$). The above mentioned distribution seemed to perform slightly better than the other two. In Figure 4.1 we have shown the numerical results on which we base our conclusions. In that Figure we see the cost function (averaged over 5 runs) of the Greedy method with the swap neighborhood using the three probabilities given above. The distribution we have selected corresponds to the green line which can be seen as the line corresponding to the lowest costs in almost each iteration[5].

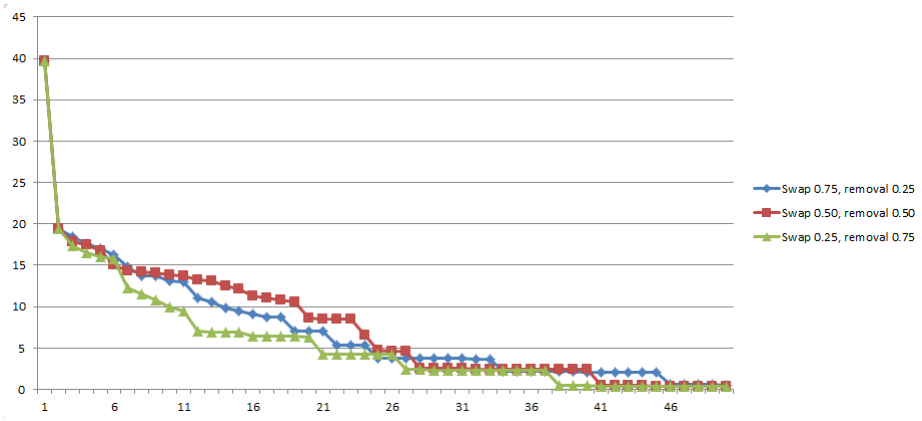

Figure 4.1: Cost function of the Greedy method with the swap neighborhood. Scenario: the basic traffic scenario. Runs: 5.

4.3 Acceptance criterion

The third aspect of our local search method is the acceptance criterion. In Section 2.2 the acceptance criterion was defined as a rule used to decide if the move to the evaluated neighbor of the current state should be made. In that same section several examples were provided of typical acceptance criteria. We have implemented two different acceptance criteria:

1. A neighbor is only accepted if it has a cost at most that of the current state.

2. A neighbor is accepted if it has a cost at most that of the current state. If the best state evaluated meets the KPI requirements we can also temporarily accept a state where the criterion will be as follows. If state x_i

[5]Note that these results use a different cost function from the one mentioned in this chapter. The cost function used in these runs was based on the load per cell. A decrease in cost is, however, still an improvement. Therefore we can also compare the methods with this cost function.

was feasible we temporarily accept the neighbor as state x_{i+1}. The neighbor of the temporary state x_{i+1} is then accepted if it has a cost lower than that of state x_i. If the neighbor of temporary state x_{i+1} is not accepted we resume our process with $x_{i+2} = x_i$.

The first acceptance criterion is used in what we call our greedy search method. The greedy in 'greedy search method' thus refers to the greedy nature of the acceptance criterion. The second acceptance criterion is a variation upon our greedy acceptance criterion which aims to provide some extra freedom in the situation where a feasible state has already been found and we are looking to reduce the financial cost. The thorough reader will notice that the goal of the second criterion is the same goal as that of the Swap operator. We will therefore never use both in the same method. In Section 4.2.4 we have mentioned our second method called the 2-step approach. The 2-step approach uses the second acceptance criterion and its neighborhood structure is defined by the operators Traffic filler and Small cell removal. The 2-step approach evaluates the network in between the two steps of removal and placement of an upgrade (hence the name). This extra evaluation in-between provides extra information about the effect of the removal of the upgrade on the networks performance on which the selection of the new upgrade can be based. The performance of the greedy method and that of the 2-step approach is compared in Chapter 6.

4.4 Stopping criterion

In Section 2.2 the term stopping criterion was introduced and some examples were provided. We have chosen to implement a combination of some of the examples we gave in that section. We use the following stopping criterion:

Stop if a fixed number of iterations is reached or if no improvement has been found for a predefined number of iterations.

The first number should be chosen such that if the search moves towards a good local optimum it is not interrupted before finding it, whilst on the other hand it should be low enough to keep the computation time acceptable in case we are not moving in such a direction. This number is difficult, if not impossible, to determine it depends very much on the structure of the search space. We choose it such that the maximum amount of iterations in a run leads to an acceptable running time. Our choice is to do no more than a hundred iterations. In Chapter 6 we will see that a hundred iterations is not always enough to avoid stopping promising searches prematurely.

The number of iterations without an improvement after which we stop depends on the size of the neighborhood of a state. This part of the stopping criterion should answer the question: 'after how many rejected neighbors do we conclude that we are in a local optimum?'. In Chapter 6 we will explain which stopping criterion was used.

4.5 Further research: incorporating problem zone characteristics

The local search methods as we have described them in this chapter all look for a feasible state in more or less the same way: identify the area (pixel) with

the largest problem and attempt to reduce the size of the overall problem by choosing upgrades around that area. This is reflected in both the cost function and in the neighborhood structure. The cost function is a summation over all pixels of the overload traffic metric, this can be seen as a way to describe both the amount of pixels with an overload and the average overload (in pixels with overload). To lower the cost function can thus be seen as reducing the size of the overall problem. The selection of a neighbor is based on the location of the pixel with the highest overload; upgrades that are closer to the location have a higher probability to be chosen[6]. The reasoning behind this is as follows: if in each step we reduce the overload in the pixel with the largest problem, then in the end we have reduced the entire problem. In theory this could lead to a very ineffective procedure since we have several types of upgrades all with a different radius in which they effect the problem.

Example. *Suppose we create a problem area by introducing a localized hotspot with a lot of potential micro sites very close to it and one macro site slightly further away. Suppose the hotspot requires several micro sites to be activated to deal with the extra demand, but using only one macro site would also suffice. Selecting upgrades based on the above criterion would mean that the macro site only has a very low probability to be selected (there are a lot of micro sites closer, i.e., a lot of sites with a higher probability). This would mean that the procedure is not likely to obtain the best solution of selecting the macro site, that is, we could say it is ineffective.*

We have dealt with this by introducing a distribution with which we first select the type of upgrade to be attempted. By doing so we account for the differences in performance of the upgrade types.

Up to this point we have only used information about the problem size (that is, we have used the total amount of overload traffic). The results of this thesis will form a part of the Decision Support System (DSS) developed in the SEMAFOUR project. Another part of the DSS focuses on detecting bottlenecks and characterizing them as problem zones. The definition of a problem zone as proposed in the SEMAFOUR project is the following:

Informal definition 4.5.1 (Problem Zone)**.** A problem zone is a connected area where in each pixel a certain sub-critical performance threshold is exceeded and at least in one pixel a higher, critical, threshold is exceeded.

In Figure 4.2 we illustrate this definition. In this thesis we have also used something much like a problem zone. We only consider the overload above a certain threshold and the neighborhood operators are focused on the area with the highest overload traffic (which can be seen as the area where a critical threshold is exceeded). The only characteristic we do not use is the connectedness. This characteristic could help our heuristic: suppose we have two disconnected problem zones, it might be easier to apply our search method to each of the problem zones separately and then combine the obtained solution. The local search methods we have described earlier can be easily adapted to treat separate problem zones. The only changes required are to the cost function and to

[6] In the previous sections we have discussed the operators that define the neighborhood of a state. The operators Traffic filler and Swap use a probability depending on the distance between upgrade and peak problem location. The impact of these distributions on the performance of the search methods has not been investigated.

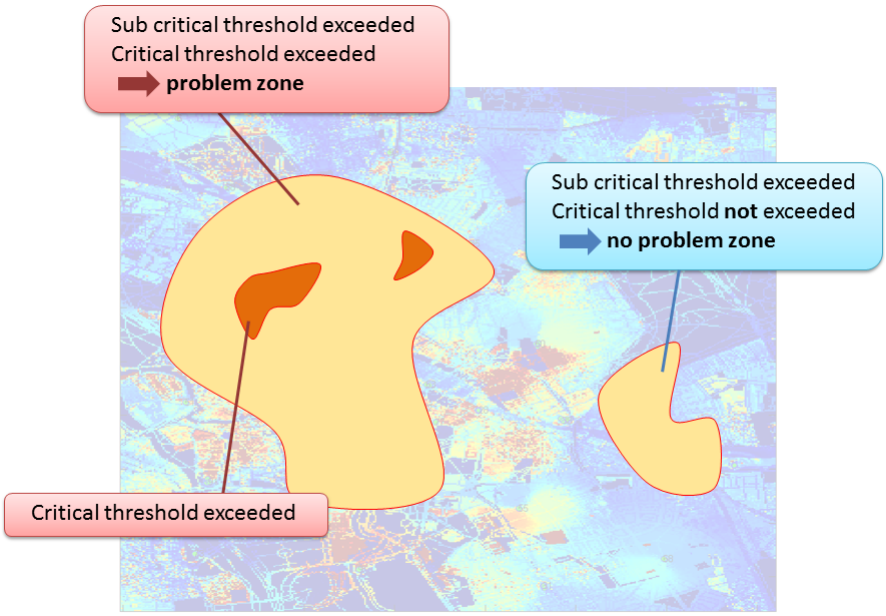

Figure 4.2: Problem zone detection. Source: Deliverable 5.3 of SEMAFOUR.

the method of selecting the area with the highest overload traffic. For both of these aspects it would suffice to limit the computations to the problem zone. The total overload in the cost function can be limited to the problem zone by limiting the summation to those pixels that lie within the problem area. As location of the highest overload traffic we can take the pixel with the highest overload traffic within the problem zone.

An important future research question is to investigate when it is beneficial to treat problem zones separately and when it is not. The reason that it can be beneficial in some cases is that, by only looking at one problem zone instead of multiple at the same time, the union of all neighborhoods seen during a run of the local search method contains fewer base stations. That means that with a fixed number of iterations we search a larger part of it. However, it can be not beneficial if the two problem zones are too close to each other. In that case it might be so that one upgrade would suffice to remove both problem zones. In that case it would not be beneficial to treat the problem zones separately.

Chapter 5

Choice of initial state

In the previous chapter we have described a local search approach to the problem of finding a network configuration which meets certain network performance standards for a predicted future traffic state. In general, a local search method can benefit from choosing a good initial state. That is, an initial state that already takes into account (some of) the structure of the problem. In this chapter we attempt to find a good initial state.

Recall the concise formulation of the main problem of this thesis (given in Section 1.3):

> Given a prediction about the future traffic demand and a list of potential network upgrades, select those upgrades that make the network meet the operators KPI requirements for the predicted traffic state, at minimum cost.

In this chapter we use the KPI load (see Appendix A.2) and we have the requirement that the load of each cell has to be below a certain threshold.

We try to find a good initial state by simplifying the above problem and formulate it as a mathematical optimization problem. First, the link is made to the capacitated facility location problem (also known as capacitated warehouse location problem or capacitated plant location problem). The capacitated facility location problem will be formulated as a binary program. We will see that the solutions of this binary program do not translate back to good solutions of the original problem. This is used as a motivation to add an extra type of constraint to the model. The resulting binary program is not easy to solve to optimality for relevant mobile network sizes, hence we consider several relaxations of the binary program. Some relaxations will not provide us with meaningful results. However, there is one promising relaxation, which is to a mixed integer linear program.

The mixed integer linear program is solved with an exact solver. However, the exact solver we use is available under the ZIB academic license, which is not meant for commercial use. Hence, we also consider a second procedure to 'solve' the mixed integer linear program. The procedure is called the dynamic slope scaling procedure and it has first been presented in [19] for the Fixed Charge Network Flow Problem (FCNFP). We will apply the same procedure to our mixed integer linear program. The dynamic slope scaling procedure is also mathematically interesting. In [19] numerical tests showed a maximum relative optimality gap of 0.65%, which is very small. The procedure was, however,

applied to a flow problem, while we consider a more general type of problem. The results obtained by the exact solver will be compared to those obtained by the dynamic slope scaling procedure.

5.1 Capacitated Facility Location Problem

The Capacitated Facility Location Problem (CFLP) is the problem of finding an assignment of lowest cost of customers (or goods) to facilities where there is a certain cost for assigning a customer to a specific facility and a fixed cost for a facility being used. This problem is well studied. In [40] an accessible overview is given of the work done until 1995. In [40] it is noted that exact methods for large instances require great computational efforts, due to the binary constraints of the values involved. Hence a smart enumeration needs to be done. Our problem

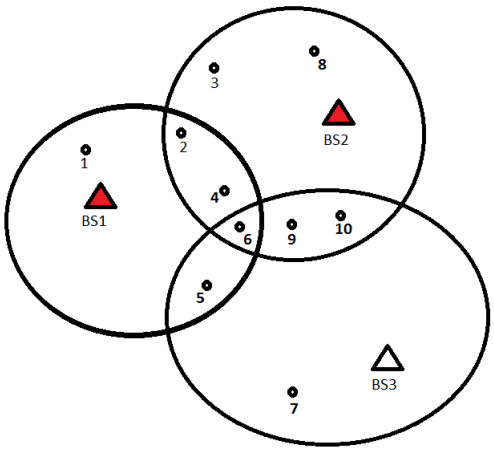

Figure 5.1: A small example of a network. The triangles denote the base stations. The ones marked red are available for 6-sectorization. The small circles denote service test points. The large circles denote the service areas of the base stations.

can be simplified and formulated as a CFLP in the following way. To have a picture in mind, consider Figure 5.1. The service test points form the set of customers, I, each with a certain demand which can be viewed as the requested traffic intensity in that area. The possible base stations form the set of facilities J. Note that on some locations we have several possible base stations (3- or 6-sector). We therefore have a set $J_{3/6} \subset J \times J$ in which each element represents a pair of a 3-sector site and 6-sector site that are on the same geographic location. We denote the use of the facilities by the variables $y_j \in \{0,1\}, j = 1,\ldots,m$ and the assignment of customer i $(i = 1,\ldots,n)$ to facility j by $x_{ij} \in \{0,1\}$. Note that we only have variables x_{ij} for the pairs (i,j) for which a connection is possible (i.e. when the received signal strength is sufficient for a connection). We therefore define the set J_i as the set of base stations that can serve STP i. Likewise, for base station j we define the set $I_j \subseteq I$ to be the set of service test points which could be connected to base station j. For each base station j we have a fixed capacity cap_j. The capacity of a base station should be seen as the

amount of traffic demand it can handle. This will be discussed in more detail in Section 5.4. The service test points i have a traffic demand d_i. Finally we have a cost c_j for the use of base station j, but we do not have a cost for the assignment of service test points to base stations. We assume that $c_j \geq 0$ for all $j \in J$. For ease of reference we have summarized the above in (5.1.1).

$$\begin{aligned}
I &\dots \{i : i \text{ is a service test point}\} \\
J &\dots \{j : j \text{ is a possible base stations}\} \\
J_i &\dots \{j \in J : \text{ STP } i \text{ can be assigned to base station } j\} \\
J_{3/6} &\dots \{(j, j') \in J \times J : \ j \text{ and } j' \text{ are possible 3- and 6-sector BSs at one location}\} \\
I_j &\dots \{i \in I : \text{ service test point } i \text{ can be served by base station } j\} \\
x_{ij} &\dots \text{ variable denoting if STP } i \text{ is assigned to BS } j \text{ (1) or not (0)} \\
d_i &\dots \text{ the traffic demand of test point } i \text{ [Mbps]} \\
y_j &\dots \text{ variable denoting if base station } j \text{ is active (1) or not (0)} \\
c_j &\dots \text{ the cost of base station } j. \text{ We assume } c_j \geq 0. \\
\text{cap}_j &\dots \text{ the capacity of base station } j \text{ [Mbps]}
\end{aligned} \tag{5.1.1}$$

Remark. In the remainder of this text an index i will be used for anything related to a service test point and the index j will be used for a base station.

The problem we arrive at can be stated as a binary program as follows:

$$\begin{aligned}
&\min \sum_{j \in J} c_j y_j \\
&\text{s.t.} \sum_{j \in J_i} x_{ij} = 1, && \forall i \in I && (5.1.2) \\
&\sum_{i \in I_j} d_i x_{ij} \leq \text{cap}_j, && \forall j \in J && (5.1.3) \\
&y_j + y_{j'} \leq 1, && \forall (j, j') \in J_{3/6} && (5.1.4) \\
&y_j \geq x_{ij}, && \forall i \in I, j \in J && (5.1.5) \\
&y_j, x_{ij} \in \{0, 1\}, && \forall j \in J, \forall i \in I_j && (5.1.6)
\end{aligned}$$

We will explain the constraints shortly. Equation (5.1.2) denotes the need of every service test point to be served by a base station. Combined with Equation (5.1.6) this tells us that all traffic from a test point is handled by a single base station. We do not allow a test point to divide, say, half its traffic to one base station and the other half to another base station. This reflects reality in the sense that the assignment of a customer is done based on the strongest received signal, so if we take a group of customers, which are in the same geographical area, together they will all be served by the same base station (as long as the area, i.e. pixel, is small enough). Equation (5.1.3) is the capacity constraint for a base station. We need the sum of all traffic assigned to the base station to be below its capacity. Equation (5.1.4) forces the problem to choose between either a 3-sector or a 6-sector site on the same geographic location (both is not a possibility in reality). Equation (5.1.5) ensures a site to be active in the case that a customer is assigned to it. Indeed, if there is an i such that $x_{ij} = 1$, then $y_j = 1$ by (5.1.5). If $x_{ij} = 0$ for all i, then (5.1.5) reduces to

$y_j \geq 0$ which holds for y_j since it is binary. Another way to model this is with a so called big-M notation. We can replace Equation (5.1.5) by $m_j y_j \geq \sum_{i \in I} x_{ij}$ where m_j is chosen as the number of customers which can be assigned to the base station. It is straightforward that this formulation has the same effect on y_j as the one stated above. The last constraint in Equation (5.1.6) forces the variables to be binary; a customer can be either assigned to a base station or not and, similarly, a base station can be active or not.

Remark. One, very important, assumption is made in Equation (5.1.3). We assume that a traffic demand at customer i has the same influence on the 'capacity' of each base station. In reality, also the distance of the customer to the base station and the interference caused by other base stations determines the effect that traffic from a customer has on the load of the base station. In fact, in Appendix A.2 we have seen that the load computation depends heavily on the received Signal to Interference plus Noise Ratio (SINR) and from Chapter 3 we know that the SINR depends heavily on both distance and interference. In Section 5.4 we discuss methods to estimate the capacity of a base station in terms of bps in traffic demand that it can serve. Recall that we want to find a good initial state for our local search methods. The quality of the initial state obtained by this binary program is largely determined by the accuracy of the capacity estimates.

5.1.1 An extra constraint: the best server constraint

In Section 5.5 we have solved a mixed integer linear program (MILP) relaxation[1] of the binary program we have presented above. The results indicate a flaw in our modeling: pixels are not necessarily assigned to the base station providing the strongest pilot signal. In practice, however, this is the case, see Appendix A. We therefore improve our model by adding the constraint that pixels have to be served by their best server. Fortunately, we can add linear constraints to our model that ensure that pixels are assigned to the correct base station. Consider a pixel i, let $(j_{i_1}, j_{i_2}, \ldots, j_{i_L}) = S(i)$ be a list of the L strongest signals received in pixel i in the order strong to weak. Define the following constraints:

$$y_{j_{i_l}} + \sum_{k=l+1}^{L} x_{ij_{i_k}} \leq 1, \qquad l = 1, 2, \ldots, L-1 \tag{5.1.7}$$

The new constraint is designed to make sure that a pixel is assigned to the first base station of the list that is active. Indeed, assume base station j_{i_h} is active, for an $h \in \{1, 2, \ldots, L-1\}$, that is $y_{j_{i_h}} = 1$. Then the h-th constraint tells us that

$$\sum_{k=h+1} x_{ij_{i_k}} \leq 0$$

and since we have $x_{ij} \geq 0$ for all existing pairs of i and j we get $x_{ij_{i_k}} = 0$ for all $k \geq h+1$. This means that pixel i cannot be served by a base station providing a weaker signal than station j_{i_h} if j_{i_h} is active. Also note that when $y_{j_{i_h}} = 0$ the constraint does not restrict our solution since each pixel can only be assigned to a total of one base station, that is, $\sum_{j \in J} x_{ij} = 1$. Therefore, adding this type of constraint results in a model in which each pixel is assigned

[1] Analogous to the MILP proposed in Section 5.2.2.

to its best server. We will call this type of constraint the best server constraint. The resulting binary program can be found below.

$$
\begin{aligned}
&\min \sum_{j\in J} c_j y_j && && (5.1.8)\\
&\text{s.t.} \sum_{j\in S(i)} x_{ij} = 1, && \forall i \in I\\
&\sum_{i\in I_j} d_i x_{ij} \le \mathrm{cap}_j, && \forall j \in J\\
&y_{j_{i_l}} + \sum_{k=l+1}^{L} x_{ij_{i_k}} \le 1, && \forall i \in I,\\
& && l = 1,2,\dots,L-1,\\
& && (j_{i_1}, j_{i_2}, \dots, j_{i_L}) = S(i),\\
&y_j + y_{j'} \le 1, && \forall (j,j') \in J_{3/6}\\
&y_j \ge x_{ij}, && \forall i \in I, j \in J\\
&y_j \in \{0,1\}, && \forall j \in J\\
&x_{ij} \in \{0,1\}, && \forall j \in J, \forall i \in I_j
\end{aligned}
$$

Note that (5.1.8) is no longer a capacitated facility location problem.

We update (5.1.1) with the above notation:

$$
\begin{aligned}
I &\dots \{i : i \text{ is a service test point}\}\\
J &\dots \{j : j \text{ is a possible base stations}\}\\
J_{3/6} &\dots \{(j,j') \in J \times J : \ j \text{ and } j' \text{ are possible 3- and 6-sector BSs at one location}\}\\
I_j &\dots \{i \in I : \text{ service test point } i \text{ can be served by base station } j\}\\
S(i) &\dots (j_{i_1}, j_{i_2}, \dots, j_{i_L}), \text{ a list of the } L \text{ strongest received signals in pixel } i.\\
&\quad \text{Ordered from strongest to weakest.}\\
x_{ij} &\dots \text{ variable denoting if STP } i \text{ is assigned to BS } j \text{ (1) or not (0)}\\
d_i &\dots \text{ the traffic demand of test point } i \text{ [Mbps]}\\
y_j &\dots \text{ variable denoting if base station } j \text{ is active (1) or not (0)}\\
c_j &\dots \text{ the cost of base station } j\\
\mathrm{cap}_j &\dots \text{ the capacity of base station } j \text{ [Mbps]}
\end{aligned}
\qquad (5.1.9)
$$

Note that the set J_i is replaced by the ordered set $S(i)$.

We have solved the binary program for the network shown in Figure 5.1 using Matlab and its function *bintprog*, the exact solution was found in under a second for this small network. For the larger instances[2] in which we are interested, Matlab was not able to give an exact solution within its standard maximum computation time of 8 hours. The purpose of this simplification is to find a good solution fast. Therefore, we are not interested in a method with a computation time of at least 8 hours. In the next section we propose relaxations of the above binary program. The aim of performing relaxations is to decrease the required computation time. A good relaxation should, however, still provide meaningful information about (5.1.8).

[2]Around 150 BSs, 1500 STPs, at most 10 possible connections for each STP.

5.2 Relaxations of the binary integer prorgam

In this section we discuss two relaxations of the binary program defined in (5.1.8). In discussing the relaxations the two important questions to keep in mind are (1) are there good solution methods[3] to this problem? (2) does the solution of the relaxation give sufficient information about the original problem?

5.2.1 Linear relaxation

The first and most natural relaxation is the following. We replace the binary constraints by the linear bounds $0 \leq y_j, x_{ij} \leq 1, \forall i \in I, j \in J$. This would transform the binary program into a linear program which can be easily solved using for example the simplex method. However, the following example will show that the solution of this relaxation no longer gives useful information.

Example (Non-integrality of the polytope)**.** *Consider a situation with one site, with the option of a 3- or* 6*-sector base station and one service test point where the demand is higher than the capacity of the* 3*-sector site. Let* y_3 *to denote the use of the* 3*-sector site and* y_6 *that of the* 6*-sector site. Let* $cap_3 = 3$ *and* $cap_6 = 6$*,* $c_3 = 1$ *and* $c_6 = 2$ *be the respective capacities and costs of the 3- and* 6*-sector base station. Let the demand of the service test point be* $d_1 = 4$*. We assume that the* 6*-sector base station provides a stronger signal to the service test point than the* 3*-sector base station. Taking the above mentioned relaxation of* (5.1.8)*, this results in the following program:*

$$
\begin{aligned}
\textit{min} \quad & y_3 + 2\ y_6 \\
\textit{s.t.} \quad & x_{13} + x_{16} = 1, \\
& 4\ x_{13} \leq 3, \\
& 4\ x_{16} \leq 6, \\
& y_3 + x_{16} \leq 1 \\
& y_3 + y_6 \leq 1, \\
& y_3 \geq x_{13}, \\
& y_6 \geq x_{16}, \\
& 0 \leq y_3, y_6, x_{13}, x_{16} \leq 1
\end{aligned}
$$

The unique optimal solution is $y_3 = 3/4, y_6 = 1/4, x_{13} = 3/4, x_{16} = 1/4$*, with cost* $5/4$*. This clearly demonstrates that the vertices of our polytope are not all integral. Translating this fractional solution back to a selection of base stations is not easy; both sites have a strictly positive value. In this small example the only feasible selection is clearly the* 6*-sector site. But in general it is not clear how a fractional solution can be rounded to an integral solution.*

From this example, we can conclude that a relaxation of all binary variables does not always lead to useful information. In the next paragraph we, therefore, only relax a subset of the binary variables.

[3] A good solution method should work well in practice. That is, for most instances provide an exact solution in reasonable time. Therefore, the simplex method would be considered good, but for instance exhaustive enumeration not. See also Chapter 2.

5.2.2 MILP relaxation

The second relaxation we consider is an option in between the original problem and the linear program proposed in the first relaxation. Starting from (5.1.8), we relax the binary constraints on the x_{ij} by replacing them with the linear bounds $0 \leq x_{ij} \leq 1$. The resulting problem is of the class mixed integer linear programs (MILP).

$$\begin{aligned}
&\min \sum_{j \in J} c_j y_j && && (5.2.1)\\
&\text{s.t.} \sum_{j \in J} x_{ij} = 1, && \forall i \in I\\
&\sum_{i \in I_j} d_i x_{ij} \leq \text{cap}_j, && \forall j \in J\\
&y_{j_{i_l}} + \sum_{k=l+1}^{L} x_{ij_{i_k}} \leq 1, && \forall i \in I,\\
& && l = 1, 2, \ldots, L-1,\\
& && (j_{i_1}, j_{i_2}, \ldots, j_{i_L}) = S(i),\\
&y_j + y_{j'} \leq 1, && \forall (j, j') \in J_{3/6}\\
&y_j \geq x_{ij}, && \forall i \in I, j \in J\\
&y_j \in \{0, 1\}, && \forall j \in J\\
&0 \leq x_{ij} \leq 1, && \forall j \in J, \forall i \in I_j
\end{aligned}$$

The Matlab version we are working with (R2012b) does not have a built in MILP solver. We therefore use an online solver. The NEOS Server [8] offers the SCIP solver[4] [11]. The SCIP solver is a non-commercial solver for mixed integer programming, its use is however restricted to research purposes under the ZIB Academic License. This means that TNO is not allowed to use the method. It is used solely for the purpose of this thesis[5]. The SCIP solver was at first unsuccessful in solving this MILP, it required a running time of more than 10 hours (which is the maximum on the NEOS Server). We therefore replaced the constraints

$$y_j \geq x_{ij}, \qquad \forall i \in I, j \in J \tag{5.2.2}$$

by the constraints

$$|I_j| \cdot y_j \geq \sum_{i \in I_j} x_{ij}, \qquad \forall j \in J \tag{5.2.3}$$

in the hope that reducing the number of constraints was sufficient to allow a quick solution with SCIP. This proved to be insufficient. The required computation time was still more than 10 hours. A final attempt to reduce the number of constraints was made by replacing the capacity constraints and the constraints in Equation (5.2.3) by the constraints

$$\sum_{i \in I_j} d_i x_{ij} \leq \text{cap}_j \cdot y_j, \qquad \forall j \in J. \tag{5.2.4}$$

[4]The CPLEX LP format [5] was used as input for the NEOS Server. In short, this means that we have written the MILP in a text file of this specific format, using the input data retrieved from SONlab.

[5]At the end of this section we mention alternatives that TNO could use.

These constraints are valid replacements. Indeed, if $y_j = 0$ then $x_{ij} = 0$ for all $i \in I_j$, as was previously imposed by constraint (5.2.3), and if $y_j = 1$ then the sum of all demand requested of j is less than its capacity. The resulting MILP can be seen below.

$$\begin{aligned}
&\min \sum_{j \in J} c_j y_j && && (5.2.5)\\
&\text{s.t.} \sum_{j \in J} x_{ij} = 1, && \forall i \in I\\
&\sum_{i \in I_j} d_i x_{ij} \leq \text{cap}_j \cdot y_j, && \forall j \in J\\
&y_{j_{i_l}} + \sum_{k=l+1}^{L} x_{ij_{i_k}} \leq 1, && \forall i \in I,\\
& && l = 1, 2, \ldots, L-1,\\
& && (j_{i_1}, j_{i_2}, \ldots, j_{i_L}) = S(i),\\
&y_j + y_{j'} \leq 1, && \forall (j, j') \in J_{3/6}\\
&y_j \in \{0, 1\}, && \forall j \in J\\
&0 \leq x_{ij} \leq 1, && \forall j \in J, i \in I_j
\end{aligned}$$

This MILP can be solved quickly by the SCIP solver. For the instance sizes we are interested in (around 150 BSs, 1500 STPs, at most 10 possible connections for each STP) the observed computation times were less than a minute; perfectly acceptable for the purpose of this simplified model.

As mentioned before, the SCIP solver cannot be used by TNO in this project. The input to the SCIP solver has been given in the CPLEX LP format [5]. The features of the CPLEX LP format we have used are reportedly also implemented in the GLPK package[6], which is a free solver, according to Gurobi [7]. As it is a free solver the GLPK package could provide an alternative for TNO. Another alternative will be presented in the next section. The MILP can be approximated using the dynamic slope scaling procedure presented in [19].

5.2.3 The equivalence of the MILP and binary program

An interesting observation we have made while solving these MILPs is that the solution was, in fact, integer for each instance. This means that we actually solved the corresponding binary program! It turns out that we can actually prove that this is always the case.

Theorem 1. *The solution of the mixed integer linear program* (5.2.5) *is in fact binary.*

Proof. The proof comes down to a simple observation. The best server constraint tells us that a pixel cannot be assigned to a base station that does not provide the strongest pilot signal. From the uniqueness of the best server we

[6] GLPK stands for GNU Linear Programming Kit. More information about GLPK can be found in [6].

therefore get that $x_{ij} = 0$ for all but one j. The first constraint, repeated below in (5.2.6), then tells us that this non-zero x_{ij} has to be equal to one.

$$\sum_{j \in J} x_{ij} = 1, \qquad \forall i \in I \tag{5.2.6}$$

That means that in a solution all x_{ij} are in fact binary. In the mixed integer linear program these are the only continuous variables, all others are binary. We can thus conclude that the entire solution vector is binary. ∎

An immediate corollary of this theorem is that the mixed integer linear program, (5.2.5), and the binary program it is a relaxation of, (5.1.8), are equivalent.

5.3 The dynamic slope scaling procedure

Our motivation to investigate the dynamic slope scaling procedure comes from the numerical results in [19]. In that article the performance of the dynamic slope scaling procedure on the fixed charge network flow problem is investigated. Unfortunately no optimality (or approximation) guarantees can be given. They did however test the procedure on problems of various sizes, for the largest problem for which an exact solution was also found (using branch and bound) the relative error was 0.65% (which was the worst relative error they found). That the largest relative error was found in the largest problem indicates that for larger problems the relative error might be worse. The computation times were very low, for the problem sizes for which they were also able to solve it exactly (around 37 nodes, 335 arcs) the computation time was about 0.1 second (whereas branch and bound took 538 seconds). For large-scale problems the computation times of course grew, but are still quite acceptable. For a problem with 202 nodes and 10200 arcs the solution was found in about one minute. Note that the amount of variables we have (roughly 15000, see Chapter 6) is comparable to the 202 nodes and 10200 arcs situation.

Remark. In [19] the dynamic slope scaling procedure was formulated and tested on the fixed charge network flow problem. This is not exactly the same problem as the model we presented in Section 5.1; we have an extra 'priority' constraint on the assignment of users to cells. It is therefore interesting to see how well the dynamic slope scaling procedure performs on the model presented in Section 5.1 compared to an exact solver such as SCIP. The results will be shown at the end of this section.

In [19] the dynamic slope scaling procedure for the Fixed Charge Network Flow Problem is presented. In this section we explain the dynamic slope scaling procedure. In particular, we first give a general explanation of the method. To illustrate the performance of the method we then present a small example. We then apply the method to the MILP presented in (5.2.5). Finally we test the procedure on several of the problem instances that we also use to test the local search methods[7].

In [19] the dynamic slope scaling procedure is formulated for the Fixed Charge Network Flow Problem, but we will see that the formulation of the

[7] These problem instances will be defined in Chapter 6.

procedure can be done without using any properties of a flow problem[8]. It can be formulated for any mixed integer linear program where the integer variables are $0-1$. As such we will state the method for a general mixed integer linear program with the added restriction that all variables are bounded by zero and one (see Section 2.4). To recall, a general mixed integer linear program, MILP, has the following structure:

$$\begin{aligned} \underset{x,y}{\text{minimize}} \quad & c_1^T x + c_2^T y \\ \text{subject to} \quad & A\begin{pmatrix} x \\ y \end{pmatrix} \leq b \\ & x, y \geq 0 \\ & x \in \mathbb{R}^{n_1} \\ & y \in \mathbb{Z}^{n_2} \end{aligned} \tag{5.3.1}$$

where $c_1 \in \mathbb{R}^{n_1}, c_2 \in \mathbb{R}^{n_2}, A \in \mathbb{R}^{m \times (n_1+n_2)}$ and $b \in \mathbb{R}^m$. We replace the last constraint by $y \in \{0,1\}^{n_2}$ to obtain:

$$\begin{aligned} \underset{x,y}{\text{minimize}} \quad & c_1^T x + c_2^T y \\ \text{subject to} \quad & A\begin{pmatrix} x \\ y \end{pmatrix} \leq b \\ & x, y \geq 0 \\ & x \in \mathbb{R}^{n_1} \\ & y \in \{0,1\}^{n_2} \end{aligned} \tag{5.3.2}$$

In Algorithm 2 we present the pseudo code of the dynamic slope scaling procedure applied to a MILP with all its integer variables binary, i.e., to (5.3.2). The method will be explained afterwards.

[8] For now we do not need to clarify what the Fixed Charge Network Flow Problem, FCNFP, is, in the following paragraph we will briefly introduce it and use it as an example to illustrate the dynamic slope scaling procedure.

Algorithm 2. *Dynamic slope scaling procedure.*

Data:
$c_1 \in \mathbb{R}^{n_1}, c_2^0 \in \mathbb{R}^{n_2}, A \in \mathbb{R}^{m\times(n_1+n_2)}$ *and* $b \in \mathbb{R}^m$.

Initialization:
Solve (5.3.2) *with the above data and the constraint* $y \in \{0,1\}^{n_2}$ *replaced by* $y \in [0,1]^{n_2}$. *Let* x^0, y^0 *be the obtained solution vector.*
From now on we denote by x^n, y^n *and* c^n *the n-th iterates of respectively the solution and cost vectors.*
Set $k := 1$

Main step:
while *stopping criterion is not fulfilled* ***do***
for $i = 1 : n_2$ ***do***

$$\left(c_2^k\right)_i = \begin{cases} \frac{\left(c_2^0\right)_i}{y_i^{k-1}} & \text{if } y_i^{k-1} > 0 \\ M_i^k & \text{if } y_i^{k-1} = 0 \end{cases}$$

where $M_i^k = \max\{\left(c_2^l\right)_i : 0 \le l \le k-1\}$.
end for
Solve

$$\begin{aligned} \underset{x,y}{\text{minimize}} \quad & c_1^{\mathsf{T}} x + (c_2^k)^{\mathsf{T}} y \\ \text{subject to} \quad & A\begin{pmatrix} x \\ y \end{pmatrix} \le b \\ & x \ge 0 \\ & 0 \le y \le 1 \end{aligned}$$

Let x^k, y^k *be the obtained solution vectors.*
$k := k + 1$;
end while

Output:
The final solution vectors x^k, y^k.

The dynamic slope scaling procedure can be characterized by three aspects: the initialization, the main step and the stopping criterion. We will discuss them in this order.

The **initialization** of the dynamic slope scaling procedure comes down to solving (5.3.2) with the input data and the constraint $y \in \{0,1\}^{n_2}$ replaced by $y \in [0,1]^{n_2}$. This means that in the first LP that we solve we use c_2^0 as the cost vector for the variables y that are intended to be binary. This is also an option mentioned in [19]. The fixed charge network flow problem allows the cost of flow f on an arc to be of the form

$$c(f) = \begin{cases} a + b \cdot f & \text{if } f > 0 \\ 0 & \text{if } f = 0 \end{cases}$$

where $a, b \in \mathbb{R}$. The initializations of the dynamic slope scaling procedure that are considered in [19] are the following. Take either $c_2^0 = \frac{a}{f_{\max}}$ or $c_2^0 = b$. If we look back at (5.3.2) then we see that we have $b = 0$. Taking the initial cost of all base stations equal to zero does not make sense, it would ignore all differences between types of base stations. By noting that $f_{\max} = 1$ in (5.3.2) we can see that our initialization corresponds to the one proposed in [19].

The **main step** in the dynamic slope scaling procedure is the update of the cost function. For the variables y_i (that is, those intended as binary variables) we check if the LP-optimal solution in the previous iteration, y_i^{k-1}, was positive. If so, we redefine the cost of that variable for the next iteration. The update is done in such a way that if y_i^{k-1} was very small (but positive), then the cost in the next step will be very high (recall that we assume that all costs are positive). In the fixed charge network flow problem this means that if an arc has a high startup cost (a fixed charge) then a small flow over that arc has a high cost per unit in the next iteration, the intended result is that this small flow will be distributed over other arcs that still below their capacity.

In the main step there is also the number M_i^k. In principal any large number for M_i^k could work. In [19] two empirical updating schemes for M_i^k are discussed:

1. M_i^k is equal to the highest cost (of that variable) used in previous iterations, i.e., $M_i^k = \max\{\left(c_2^l\right)_i : 0 \leq l \leq k-1\}$.

2. M_i^k is equal to $\left(c_2^l\right)_i$ where $l < k$ was the last iteration in which $y_i^l > 0$ (with $M_i^k = \left(c_2^0\right)_i$ if such an iteration does not exist).

It is noted that both schemes work well on most problems they tested, but for problems which required a large number of iterations (empirically more than a hundred) the last one performed slightly better. We have implemented the first, because we do not use a large number of iterations.

A **stopping criterion** needs to be determined as well. In [19] a very natural stopping criterion is used: stop when a fixed point is reached. Since we do not want to get stuck in an infinite loop we add a restriction on the number of iterations. This idea is very similar to the stopping criterion we have used in our local search methods, see Section 4.4. In Section 5.5 we will present a numerical comparison between the exact solution of the MILP and the dynamic slope scaling procedure. At that point we will comment on the number of iterations used.

5.3.1 Example: dynamic slope scaling procedure applied to a network flow problem

The workings of the dynamic slope scaling procedure can be best illustrated with a small example. In [19] the dynamic slope scaling procedure is formulated for the fixed charge network flow problem, since network flows can be easily visualized we have chosen to use it as an example.

Let us give a brief introduction to the network flow problem. A network flow problem is the problem of finding flow values for the arcs of a directed graph such that in each node the demand is met, there is conservation of flow and the total cost of the flow is minimized. The fixed charge network flow problem

allows the cost of flow f on an arc to be of the form

$$c(f) = \begin{cases} a + b \cdot f & \text{if } f > 0 \\ 0 & \text{if } f = 0 \end{cases}$$

where $a, b \in \mathbb{R}$. The constant a explains the 'fixed charge' part of the name, it is a fixed charge required to use a specific arc (it can be seen as a startup cost of sorts). The arc has a certain capacity $f_{\max}$.

We will now present an example fixed charge network flow problem to which we have applied the dynamic slope scaling procedure. We consider a situation with three base stations and three service test points. The possible connections between users and base stations can be seen in Figure 5.2. The base stations all have the option to be either a 3-sector or 6-sector site. A 3-sector site has capacity[9] equal to 3 and cost equal to 1. Likewise, a 6-sector site has capacity equal to 6 and cost equal to 3. The 6-sector site has double the capacity and thrice the cost of a 3-sector site, hence we favor a solution with as much 3-sector sites as possible. The demand vector b of the service test points is shown above the nodes, respectively $1, 2$ and 2. A quick calculation shows that the total demand is low enough to, in theory, be handled by two 3-sector sites (which would be the solution of lowest cost). It is easy to verify that by using base stations 2 and 3 a feasible solution of this cost exists (assign STP1 and STP2 to BS2 and STP3 to BS3). We have not explained yet how we would be able to handle the choice between a 3- or 6-sector base station at the same site, this can be done using the construction shown in Figure 5.3, this would yield the actual network flow problem denoted in Figure 5.4. As we can see Figure 5.4 is quite complicated, therefore we use Figure 5.2 for the visualization in the remainder of this example.

In this example we would like to show if and how the dynamic slope scaling procedure would find a solution of lowest costs, starting from a different flow (with higher cost). For this we need to define a different, sub-optimal, flow. In Figure 5.4 such a flow is given by the uninterrupted arcs (the dotted arcs represent possible alternative assignments of STPs to BSs). It is not hard to see that this assignment completely determines the flow. We see that BS3 has to supply a flow of 4, which means that it is a 6-sector site, BS1 only carries a flow of 1 so it is a 3-sector site.

Iteration 1:

We define the cost function as stated above. That means, for most arcs we take the cost equal to the original cost in the network, only since BS3 carries a flow of 4 we need to modify the arcs belonging to it. The arc corresponding to the 3-sector part (see Figure 5.3) carries a flow of 3, so we divide its cost by 3 to get a cost of $1/3$. The arc corresponding to the 6-sectorization of that 3-sector site carries a flow of 1, so we should divide its cost by one. Similarly the arcs corresponding to the 3-sector site of BS1 also carry a flow of 1, so their costs should be divided by one. Solving the corresponding LP problem yields a solution in which the flow over BS3 is reduced to a value of 3. This is done by reducing the flow between BS3 and STP2 by 1, this demand of STP2 now flows through the arc (BS2,STP2). This means that the cost of the network is reduced from 4 (one 3-sector BS plus one 6-sector BS) to 3 (three 3-sector BSs).

[9] Capacity, cost and demand is used without looking at appropriate units.

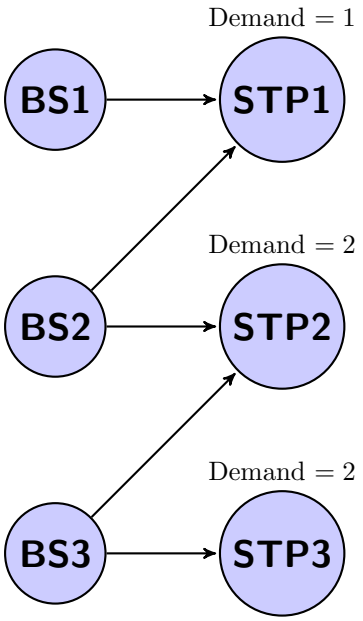

Figure 5.2: Mobile network, example FCNFP.

The arcs between base stations and service test points which carry a positive flow together with the flow value:

$(BS1, STP1)$	0.8137
$(BS2, STP1)$	0.1863
$(BS2, STP2)$	1
$(BS3, STP2)$	1
$(BS3, STP3)$	2

In Figure 5.5 we can see the results of the first iteration. The uninterrupted arcs indicate a positive flow over the arc and the red arcs indicate a change from the previous situation.

Iteration 2:

The cost function is updated based on the existing solution. The resulting LP problem is solved and provides us with the optimal solution, shown in Figure 5.6. The arcs between base stations and service test points which carry a positive flow together with the flow value:

$(BS2, STP1)$	1
$(BS2, STP2)$	1
$(BS3, STP2)$	1
$(BS3, STP3)$	2

We can recognize that this is the solution we are looking for: it is of cost 2. To show that this is indeed a fixed point of the problem we have to update the cost function again and solve the resulting LP problem. Doing this shows that the flow does not change anymore. Hence we have reached a fixed point.

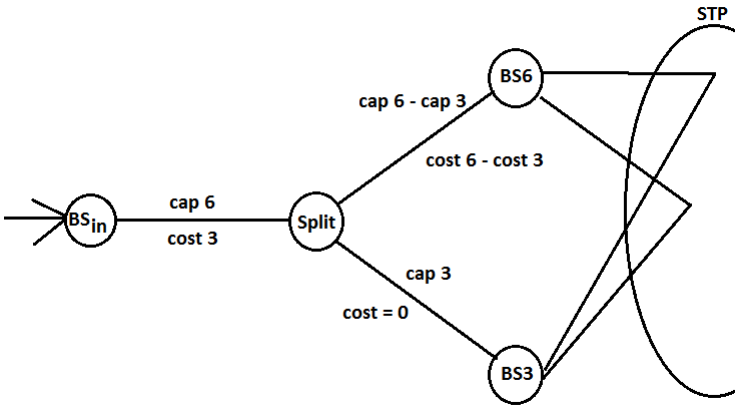

Figure 5.3: A 3- or 6-sector base station splitted with its cost and capacity defined.

Conclusion:

We can see that the dynamic slope scaling procedure was indeed able to find a minimum cost flow for this network problem. The predicted behaviour was indeed observed: we favor 3-sector base stations and, when two base stations have a enough slack to assign an STP (in our example STP1) we assign the STP to the base station with the highest flow. This resulted in being able to close one of the 3-sector base stations.

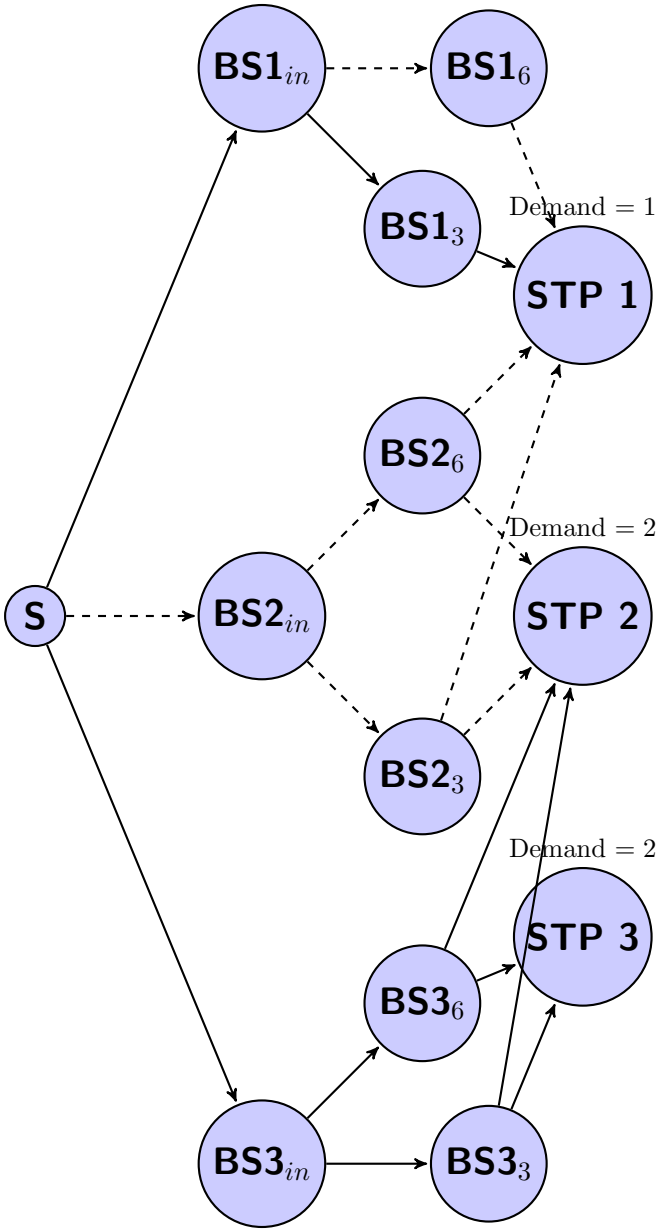

Figure 5.4: Initial state of an example FCNFP.

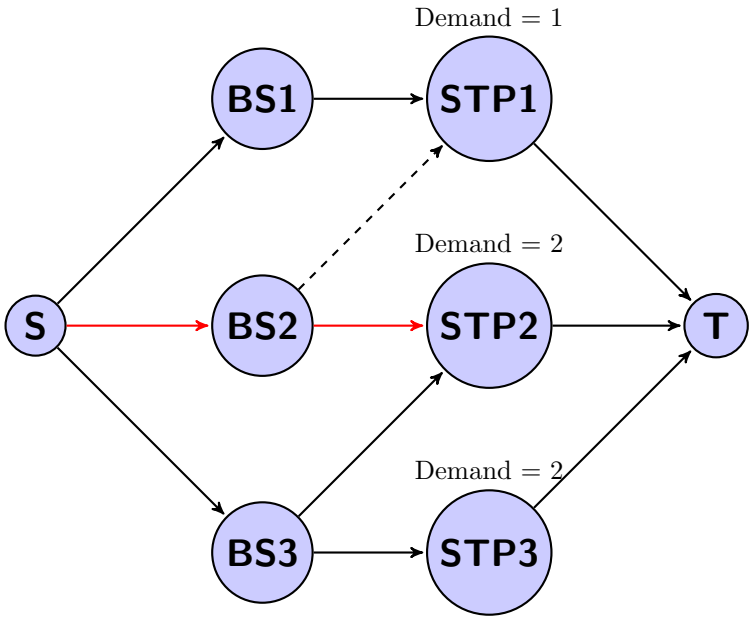

Figure 5.5: Positive flows (uninterrupted lines) and possible alternatives (dotted lines) after one iteration. In red the changes with respect to the previous solution are shown.

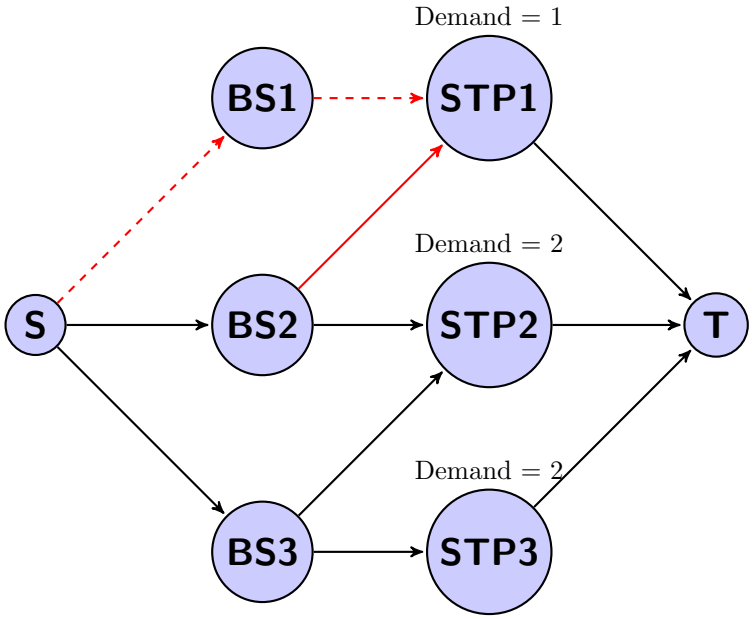

Figure 5.6: Positive flows (uninterupted lines) and possible alternatives (dotted lines) after two iterations. In red the changes with respect to the previous solution are shown.

5.4 Cell capacity estimates

We would like to define the 'capacity' of a cell in terms of bits per second of traffic demand (from the user side) we expect it to be able to handle without problems[10]. The motivation to do so should be clear from the earlier parts of this chapter; all optimization models proposed from Section 5.1 onwards depend heavily on this. Up to now, we have usually worked with the load of a cell as an indication of how 'busy' the cell was, i.e. what percentage of its physical resources was used. In Section A.2, we have discussed the load of a cell and related concepts in more detail. The physical resources of a cell are the available bandwidth in hertz. The available bandwidth is known; it is the amount of bandwidth the operator bought at the auction[11]. So, we want to find a relation between the amount of bits per second requested by a user and the amount of hertz a cell needs to supply this demand. This would give us an indication of the capacity of a cell.

In this section we present three different methods to define the capacity of a cell in terms of bits per second of traffic demand. The first method is a rough estimate based on a scenario analysis using SONlab. The second method is a very practical approach. The third method is the more theoretical one. The third method is able to provide us with a theoretical upper bound on the capacity under certain assumptions.

Before describing the three methods it is useful to recall a bit of theory, described in more detail in Appendix A.2. The theory will motivate the different approaches we take.

The relation between traffic demand and hertz required is not as clear cut as one would hope, it depends on the interference level at the users location. That is, it depends on the Signal to Interference plus Noise Ratio, SINR. The relation is via the well known Shannon formula (also known as part of the Shannon-Hartley theorem):

$$R_{bps} = B \cdot {}^{2}\log(1 + \text{SINR}) \tag{5.4.1}$$

where R_{bps} is the amount of bps a user could receive if it could use the full bandwidth B [Hz] under a given SINR. The SINR is taken as its linear value, not, as it is commonly noted, in dB. To have an idea of how the SINR (in dB) relates to R_{bps} consider Figure 5.7. In the next paragraph we will see an example where the SINR varies between 1 and 40 dB (see Figure 5.8).

The next step is to compute the fraction of the bandwidth requested by this pixel. We do this by dividing its demand D_{bps} by the theoretical maximum received data rate R_{bps}:

$$\frac{D_{bps}}{R_{bps}}$$

The load of a cell is then the sum over all pixels assigned to that cell of their required fraction of bandwidth. In Appendix A.2 we have explained that the load of a cell influences the SINR received at a pixel and hence the load computation is in fact a fixed point problem. The binary program formulation in this section circumvents this fixed point problem by assuming a fixed influence of

[10] For us, without problems means that the load of each cell remains below the threshold of 0.6.

[11] In this thesis a bandwidth of 10 MHz is used.

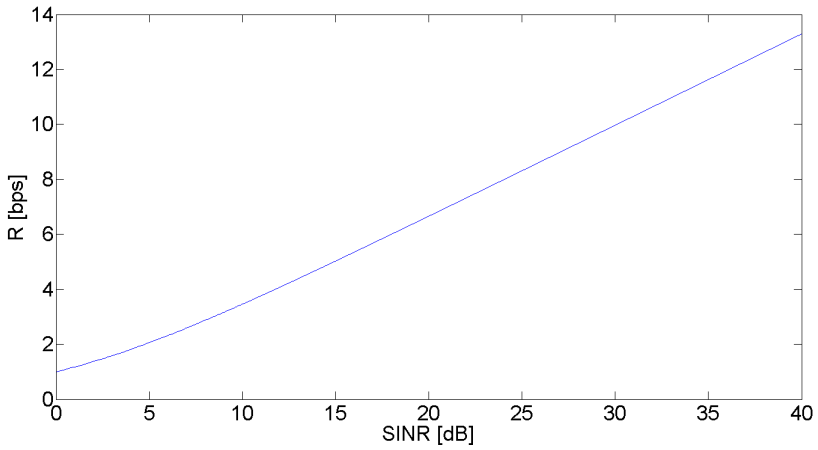

Figure 5.7: A plot of the relation between SINR [dB] and R [bps] given in Equation (5.4.1). We use $B = 1$.

the traffic demand of a pixel on the load of a cell. This implicitly means that we are making assumptions on the level of interference at each pixel. In Subsection theoreticalbounds we will make these implicit assumptions explicit in order to provide theoretical bounds on the capacity (under certain assumptions).

5.4.1 Method 1: rough estimate using SONlab

Our first method was used to get a rough estimate of the order of magnitude we should look for. The method was very simplistic, we considered one scenario (the same as in Section 5.5). We took the sum of all traffic offered and divided it by the sum of the load of each cell. The total traffic offered was 93.97 Mbps, the sum over all loads was 17.02. We want to note that we took the offered traffic such that the maximum load was only slightly below our threshold of 0.6 and realistically distributed. The relative intensity map was designed within the SEMAFOUR project using methods described in [18], as such we will not discuss whether it is a realistic scenario or not, we just assume so.

From the numbers given above we can come up with a traffic demand per cell which would result in a load of 1 (i.e. all resources are used):

$$\frac{93.97}{17.02} = 5.52\text{Mbps}$$

since we do not allow a load of 1 but instead take a maximum load of 0.6 this would give us a capacity per cell of $0.6 \cdot 5.52 = 3.3$ Mbps.

It is very important to note that by doing this division we implicitly assume that the load of one cell is independent of the load of another cell. In reality this is not the case since a higher load of a neighboring cell causes a higher interference level the cell and thereby a higher load of the cell. Another way to look at it is that by increasing the demand in one cell we also raise the load of all neighboring cells to a certain extent. A second assumption we implicitly

make in this method is that on average the fractions of demand experiencing certain SINR levels are the same for each cell. Both of these assumptions are rather big (for the second one, see for example Figure 5.8). This motivates us to call this method a rough estimate.

5.4.2 Method 2: trial-and-error

The second method is a very practical approach. In the end of this section we will discuss the three methods and the need for this method will become apparent. This method is also based on the scenario as described in Section 5.5, for this scenario we have determined a scaling factor which gave us a maximum load slightly below the threshold. This means that we know that the MILP applied to this scenario should have the existing network as feasible (and hence optimal) configuration, while lowering the capacities slightly would result in upgrades to be selected.

The method gives us a capacity per cell of 2.09 Mbps. Note that this is roughly two thirds of the capacity obtained using method 1. An explanation of this difference is the following. In our MILP (5.2.5) we take each base station as a variable, also pixels are assigned to base stations instead of cells. This means that in our MILP we use a capacity per base station which is the amount of cells of that base station times the capacity of a cell. If traffic is distributed homogeneously this is the same. By traffic being distributed homogeneously we mean that in the actual network each cell of a base station would have the same load. In that case exceeding the capacity of a base station would mean that each cell of that base station exceeds its capacity. Hence the obtained capacity is indeed the correct estimate. However, if traffic is not distributed homogeneously it could be the case that traffic that is in reality assigned to cell 1 of a base station is now handled using the capacity of, say, cell 2. The following example shows that if the load is not the same for each cell of a base station, then we find an underestimate of the capacity per cell.

Example. *Take a base station with two cells. Let the capacity of each cell be* 1*. The total offered traffic to cell* 1 *is* 1 *and to cell* 2 *is zero. Then, in our MILP* (5.2.5)*, it would suffice to take a capacity for the base station of* 1*. Which means a capacity per cell of* 1/2*. Since then all demand can be supplied (*1/2 *by cell* 1*,* 1/2 *by cell* 2*).*

5.4.3 Method 3: theoretical bounds

We have mentioned earlier Shannon's formula which relates the SINR to a theoretical maximum amount of bits per second which can be received from a cell. This theoretical maximum however excludes the need for all kinds of overhead in the system (cyclic prefixes, overhead caused by the pilot signal, etc.). In [39] these kinds of overhead are studied for LTE they propose a correction factor η where $\eta \approx 0.6$. We would also like to note that our base stations use a 2x2 MIMO configuration. This will also be noted in Chapter 6 when we discuss the scenario, but it is useful to take it into account in this section as well. MIMO stands for Multiple-Input Multiple-Output. In short[12], MIMO can be described

[12]For more general information about MIMO, the reader could for instance look at the (english!) wikipedia page on MIMO. It gives an accessible description of the MIMO concept.

as a way of configuring your antenna array's for each sector in such a way that you get two more or less separated channels between sender and receiver (who also has two antenna's, hence the last two in 2x2 MIMO). In [39] the impact of the MIMO configuration is also mentioned. Important for us is that this doubling of channels means that the cell capacity should also be doubled. This all leads us to a corrected Shannon bound, which we will denote by $S_{\max}$, of the following form:

$$S_{\max} = 2 \cdot \eta \cdot B \cdot {}^2\log(1 + \text{SINR}) \tag{5.4.2}$$

where η is the correction factor proposed in [39], B is the available bandwidth and SINR is used in its linear form. For later use we would like to define

$$C_{\max} := 0.6 \cdot S_{\max}$$

as our capacity including our wish to not exceed a load of 0.6.

Now let us consider $S_{\max}$ as a function of the SINR. We can clearly see that it is an increasing function of the SINR. This observation will play a key role. If we can give an upper bound on the SINR then we automatically get an upper bound on the $S_{\max}$ and thus on $C_{\max}$. If we take for instance a maximum SINR of 40 dB we get

$$C_{\max} = 0.6 \cdot 2 \cdot 0.6 \cdot 10^6 \cdot {}^2\log(1 + 10^{\frac{40}{10}}) \approx 9.56[\text{Mbps}].$$

This bound is quite high, but it assumes that each pixel has an SINR of 40 dB, which is not realistic. See for example Figure 5.8, the SINR map of the scenario used in methods 1 and 2. A better bound can be found by using smaller intervals to which the SINR of the traffic demand belongs. We can then take the upper bounds of these intervals to get a $C_{\max}$ per interval and then average the results in a good way. Abusing notation a little bit we denote by $C_{\max}$ (SINR) the capacity we would get if all our bandwidth served pixels with a certain SINR. This leads us to a better bound of the form shown below.

Definition 5.4.1 (Weighted Corrected Shannon Bound)**.** Let $\mathcal{I} = \{I_1, I_2, \ldots, I_k\}$ be a set of intervals with disjoint interior and such that for all $x \in I_i, y \in I_{i+1}$ we have $x \leq y$ $(i = 1, 2, \ldots, k-1)$. Furthermore let SINR_i be the supremum of interval I_i for $i = 1, 2, \ldots, k$. We define $C^*(\mathcal{I}, \lambda)$, the Weighted Corrected Shannon Bound, as:

$$C^*(\mathcal{I}, \lambda) = \sum_{i=1}^{k} \lambda_i \cdot C_{\max}(SINR_i) \tag{5.4.3}$$

where $\lambda_i \geq 0$ for all i and $\sum_{i=1}^{k} \lambda_i = 1$.

The following two examples should clarify this bound.

Example (Weighted Corrected Shannon Bound)**.** *Suppose that of the traffic demand assigned to a cell* 20% *experiences an SINR below* 10 *dB,* 40% *between* 10 *and* 20 *dB,* 20% *between* 20 *and* 30 *dB and the remaining* 20% *between* 30 *and* 40 *dB. The weighted bound given in Equation* (5.4.3) *then gives us the following upper bound:*

$C_{\max} = 0.2 \cdot C_{\max}(10) + 0.4 \cdot C_{\max}(20) + 0.2 \cdot C_{\max}(30) + 0.2 \cdot C_{\max}(40) = 5.76$ *Mbps*

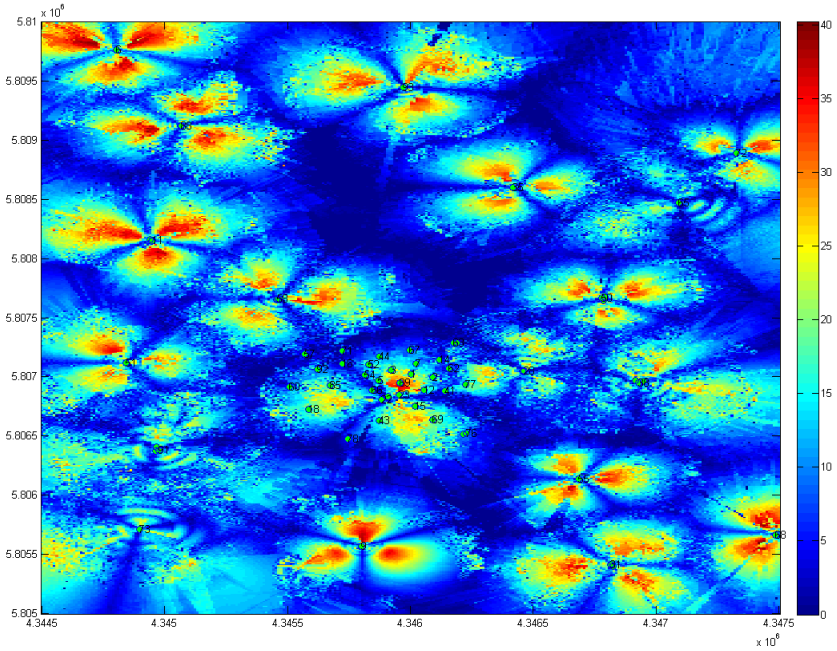

Figure 5.8: The SINR map of the 3x5km^2 Hannover region using the basic scenario with scaling factor 0.026.

Example (Weighted Corrected Shannon Bound, Example 2)**.** *Suppose that of the traffic demand assigned to a cell* 50% *experiences an SINR below* 10 *dB,* 25% *between* 10 *and* 20 *dB,* 20% *between* 20 *and* 30 *dB and the remaining* 5% *between* 30 *and* 40 *dB. The weighted bound given in* (5.4.3) *then gives us the following upper bound:*

$$C_{\max} = 0.5{\cdot}C_{\max}(10){+}0.25{\cdot}C_{\max}(20){+}0.2{\cdot}C_{\max}(30){+}0.05{\cdot}C_{\max}(40) = 4.35 \text{ } \textit{Mbps}$$

The importance of the Weighted Corrected Shannon Bound will become clear from the next two theorems.

Choosing the best weights and categories in the Weighted Corrected Shannon Bound is not an easy task. Even for a given set of intervals choosing the weights in an optimal way is difficult. The optimal weights depend on the distribution of traffic amongst cells, these are not known until a network configuration is chosen (which is the output of the MILP). What helps however is the following:

Theorem 2. *Take a fixed set of intervals $\mathcal{I}$ as in Definition 5.4.1 and weight sets $\{\lambda_i\}_{i=1}^k$ and $\{\lambda_i'\}_{i=1}^k$ satisfying $\lambda_i, \lambda_i' \geq 0$ for all i and $\sum_{i=1}^k \lambda_i' = \sum_{i=1}^k \lambda_i = 1$. If for each $n \in \{1, 2, \ldots, k\}$ we have:*

$$\sum_{i=1}^{n} \lambda_i \leq \sum_{i=1}^{n} \lambda_i' \tag{5.4.4}$$

then the following inequality holds: $C^*(\mathcal{I}, \lambda) \geq C^*(\mathcal{I}, \lambda')$.

Intuitively this theorem can be viewed as follows: the weighting λ is a more optimistic view of the situation than the weighting λ', in terms of the SINR received by a user.

Proof. As we have noted before $C_{\max}(\text{SINR})$ is an increasing function of the SINR. The intervals are fixed and satisfy a certain ordering: for all $x \in I_i, y \in I_{i+1}$ we have $x \leq y$ $(i = 1, 2, \ldots, k-1)$. The desired inequality now follows by a combination of the above with inequalities 5.4.4. We start with $C^*(\mathcal{I}, \lambda')$.

$$\begin{aligned} C^*(\mathcal{I}, \lambda') &= \sum_{i=1}^{k} \lambda_i' \cdot C_{\max}(\text{SINR}_i) \qquad (5.4.5) \\ &= \sum_{i=1}^{k} \lambda_i \cdot C_{\max}(\text{SINR}_i) + \sum_{i=1}^{k} (\lambda_i' - \lambda_i) \cdot C_{\max}(\text{SINR}_i) \\ &= C^*(\mathcal{I}, \lambda) + \sum_{i=3}^{k} (\lambda_i' - \lambda_i) \cdot C_{\max}(\text{SINR}_i) + \sum_{i=1}^{2} (\lambda_i' - \lambda_i) \cdot C_{\max}(\text{SINR}_i) \\ &\leq C^*(\mathcal{I}, \lambda) + \sum_{i=3}^{k} (\lambda_i' - \lambda_i) \cdot C_{\max}(\text{SINR}_i) + \sum_{i=1}^{2} (\lambda_i' - \lambda_i) \cdot C_{\max}(\text{SINR}_2) \end{aligned}$$

In the last inequality we use Equation (5.4.4), this gives us $\lambda_1' - \lambda_1 \geq 0$ and the inequality $C_{\max}(\text{SINR}_1) \leq C_{\max}(\text{SINR}_2)$. This step can be generalized and repeated (induction!). We use the following:

$$\begin{aligned} \sum_{i=1}^{n} (\lambda_i' - \lambda_i) C_{\max}(\text{SINR}_n) + (\lambda_{n+1}' - \lambda_{n+1}) C_{\max}(\text{SINR}_{n+1}) \\ \leq \sum_{i=1}^{n+1} (\lambda_i' - \lambda_i) C_{\max}(\text{SINR}_{n+1}) \end{aligned}$$

where we use Equation (5.4.4) to get $\sum_{i=1}^{n} (\lambda_i' - \lambda_i) \geq 0$ and the inequality

$$C_{\max}(\text{SINR}_n) \leq C_{\max}(\text{SINR}_{n+1})$$

which holds for all $n = 1, 2, \ldots, k-1$ since $C_{\max}(\text{SINR})$ is an increasing function of SINR.

So from inequality (5.4.5) we get using the argument above:

$$C^*(\mathcal{I}, \lambda') \leq C^*(\mathcal{I}, \lambda) + \sum_{i=1}^{k} (\lambda_i' - \lambda_i) C_{\max}(\text{SINR}_k) \qquad (5.4.6)$$

The last step we need to make is the following observation. For λ and λ' we have the following equality

$$\sum_{i=1} \lambda_i' = \sum_{i=1} \lambda_i (= 1).$$

So in Equation (5.4.6) the right most term vanishes. This gives us the inequality we need:

$$C^*(\mathcal{I}, \lambda') \leq C^*(\mathcal{I}, \lambda)$$

■

Now that we have Theorem 2 we can prove the main result of this method.

Theorem 3 (Lower bound)**.** *Consider a traffic scenario and a set of intervals $\mathcal{I}$. Let λ be an optimistic weighting as in Theorem 2, that is a weighting for which we know that the actual weighting λ' in each potential cell satisfies Theorem 2. Then the solution of the MILP formulated in Section 5.1,*(5.2.5)*, using cell capacity $C^*(\mathcal{I}, \lambda)$, forms a lower bound to the solution of the problem described in Section 1.3.*

Proof. $C^*(\mathcal{I}, \lambda)$ is an overestimate of the capacity in bits per second per cell. Now take a feasible network configuration for the problem defined in Section 1.3, that is a network in which the maximum load of a cell is 0.6. Consider the same network configuration in the MILP, with the same assignment of pixels to cells. The choice of $C^*(\mathcal{I}, \lambda)$ and the fact that it is an overestimate of the capacity of each cell show that each cell in the network configuration does not exceed its capacity in the MILP. Furthermore SONlab also assigns pixels to the cell providing the strongest signal. Hence each feasible solution of the problem defined in Section 1.3 is also feasible for the MILP. Since the MILP is a minimization problem this shows that the solution to the MILP is a lower bound to the problem defined in Section 1.3. ∎

The strength of this theorem lies in the ability of the network designer to be able to choose such an optimistic weighting λ. This is however also the greatest drawback to this method; the network designer has to choose such a weighting. can be used in the MILP to provide us with a lower bound on the optimal value of the problem presented in Section 1.3, under certain If we, for example, assume the traffic distribution in all cells will satisfy the distribution mentioned in the first example of this paragraph, then we can use a capacity of 5.76 Mbps in our MILP. Theorem 3 then shows that the solution of the MILP will be a lower bound to the problem defined in Section 1.3.

5.4.4 Conclusion

In this section we have seen three methods to estimate the capacity of a cell. The first method requires assumptions that can hardly be justified. The third method improves upon the first by allowing us to make more reasonable assumptions. The third method allows one to make a trade-off between the strength of the assumptions and the quality of the estimate. By making more detailed assumptions the estimate of the cell capacity becomes better, but making detailed assumptions requires a lot of knowledge about the distribution of traffic and the cell lay-out. A big advantage of the third method is that it does give a guaranteed upper bound on the cell capacity, but under certain assumptions. The second method also makes certain assumptions just like the first method. However, it does so at a base station level instead of a cell level. Moreover, the second method is the only method that explicitly takes into account our modeling of the network configuration in terms of base stations instead of cells. A final advantage of the second method over the third method is that it is not vulnerable to a change of units of the offered traffic. The last point is important since, as we have mentioned in Chapter 3, we are not sure about the units used in the traffic intensity map. We treat them as if they are Mb's but they could also be 2Mb's for all we know.

In Chapter 6 we have chosen to use our estimate obtained via the second method.

5.5 Numerical examples

In this section we will present numerical examples for the methods and formulations we have seen in this chapter. Where possible we will use the same scenario.

We start with an illustration of the performance of the original binary program, i.e., the binary program without the best server constraint. To show that the best server constraint has the desired effect we will then illustrate the performance of the model with this constraint. To that same scenario we will apply the dynamic slope scaling procedure. The dynamic slope scaling procedure will show to be unsuccessful in solving the MILP.

5.5.1 The MILP without the best server constraint

The scenario we use is the 3×5 km^2 Hannover region presented as the basic scenario in Appendix B.2. The traffic intensity grid has a few outliers in terms of traffic demand (pixels with a demand around 1). This makes it difficult to see the pattern clearly. In Figure 5.9 we have replaced all data points by the minimum of 0.1 and the original data point. The bottleneck in this scenario is a load of 0.814 for cell 43 belonging to site 22, all other sites have a load below the threshold of 0.6. In the examples in this section we have used the following

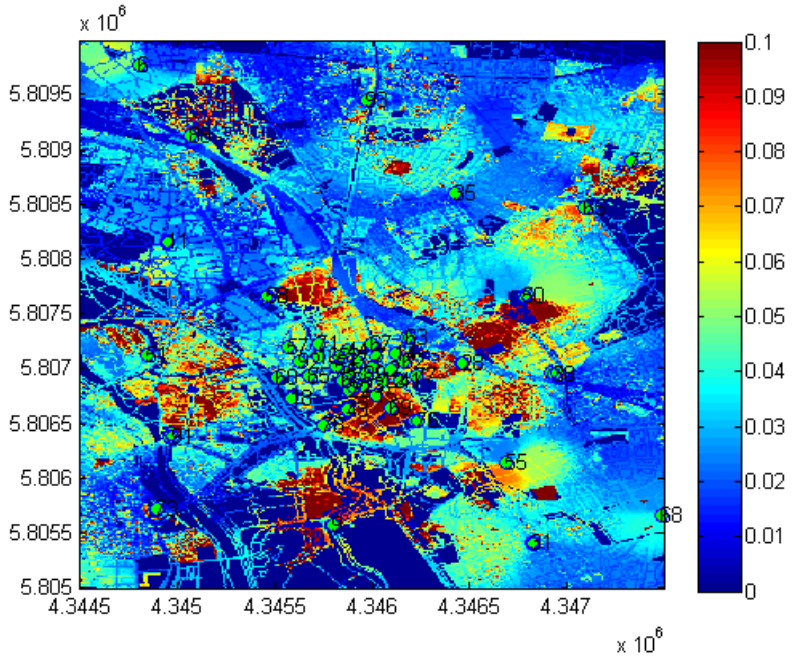

Figure 5.9: The traffic intensity grid capped at 0.1 on the 3×5 km^2 Hannover region.

costs:

1. the existing network is assumed to have cost zero,
2. performing a 6-sectorization on a 3-sector site has cost one and,
3. building a new micro site has cost 5.

The capacity of the sites is chosen as $C \cdot 3, C \cdot 6$ or C for respectively a 3-sector, 6-sector or micro site. The constant C will be referred to as the *capacity scaling* constant. In Section 5.4 we took a closer look at the capacity scaling constant, when applicable we will use the results of this section.

In Section 5.1 we have proposed a binary program in Equations (5.1.2) through (5.1.5). This binary program was relaxed to a MILP in the same way as in SubSection 5.2.2.

The solution of the MILP, using capacity scaling constant $C = 0.72$, has been visualized in Figure 5.10. The selected upgrades are a 6-sectorization of sites 65 and 73. The evaluation of the chosen upgrades with SONlab showed that the maximum load of a cell in will be 0.805, which is a negligible improvement of the original bottleneck of 0.814 (the cell with the maximum load remains unchanged).

We can see in Figure 5.10 that the selected upgrades are not exactly where we would expect. Both upgrades are selected far from a region with high traffic. In fact, there are other (active) 3-sector sites closer to the high traffic intensity regions than the ones we selected. As we know a pixel should be assigned to a base station providing the strongest pilot signal. This means that the sectorization of sites 65 and 73 does not solve the bottleneck at site 22 since site 22 provides a stronger signal to the areas with high traffic intensity.

In Section 5.1.1 we therefore added a constraint to the binary program to ensure that a pixel is served by its best server, this resulted in the binary program seen in (5.1.8).

5.5.2 The MILP with the best server constraint

In Section 5.2.2 we presented the mixed integer linear program relaxation of the binary program with the best server constraint. This MILP can be seen in (5.2.5).

The solution of the MILP using the same capacity scaling constant ($C = 0.72$) as in the previous paragraph resulted in 12 sites to be 6-sectorized. This is not comparable to the amount of upgrades selected in the previous paragraph[13].

We have applied (5.2.5) to one of our scenarios that has very clear hotspots, namely instance 7 of Chapter 6. The capacity scaling constant we used is the same as the one we found with method 2 of Section 5.4: 1.09. In Figure 5.11 we can see the result. It is very clear that in this case the base stations closest to the hotspots are selected to be upgraded. For the sake of comparison we would like to mention that the computation time on the NEOS server was 3.75 seconds, although the same MILP for other instances has been seen to take up to a minute to solve. A relation between the optimal value and the computation time seems to exist; the higher the longer the required computation time. This is

[13]However, it is a logical outcome! We restrict the assignment of users to test points and thereby we force the MILP to select more upgrades.

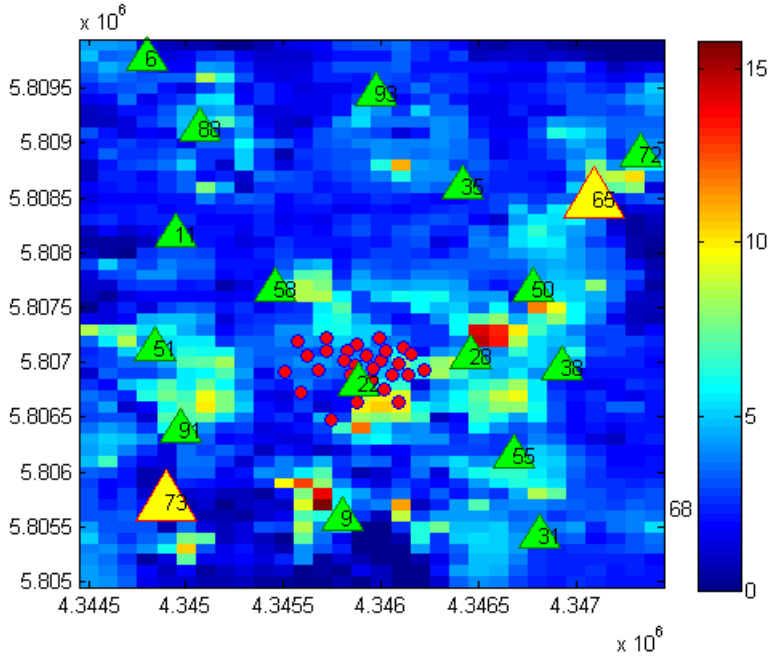

Figure 5.10: Selected BSs by the MILP relaxation of the binary program (5.1.2)-(5.1.5) plotted over the traffic intensity grid. The green triangles represent existing 3-sector sites. The red circles represent micro sites (not selected). The larger yellow triangles are the upgrades selected by the MILP.

likely because more upgrades need to be selected. Further testing could confirm this, but since computation times of around a minute are perfectly acceptable for us we did not look into it.

5.5.3 The dynamic slope scaling procedure

In Section 5.3 we have presented the dynamic slope scaling procedure. This procedure has been applied to the same scenario as was used in the previous subsection.

We first applied it with the cost updating scheme as in Algorithm 2. The result was a very fast convergence (more on that later), but unfortunately not a useful solution. In Figure 5.12 we have shown a part of the solution. In the figure we see the results of the first five iterations for the variables corresponding to the first thirty macro base stations (i.e., the y_i^k in Algorithm 2 for $i = 1, \ldots, 30, k = 1, \ldots, 5$). We chose this subset of the variables since it represents the (non-zero part of the) solution well; for the 63 macro 3-sector base stations the results were similar (the peak variable had a value of 0.9136), all micro base stations and 6-sector sites are not selected (values in the range of 10^{-15}, i.e., machine precision).

We have mentioned above that these results are not very useful. The reason is that we intend the base station variables to be either 0 or 1 and the solution shows them in the range of roughly 0.4 to 0.9. That means there is no logical

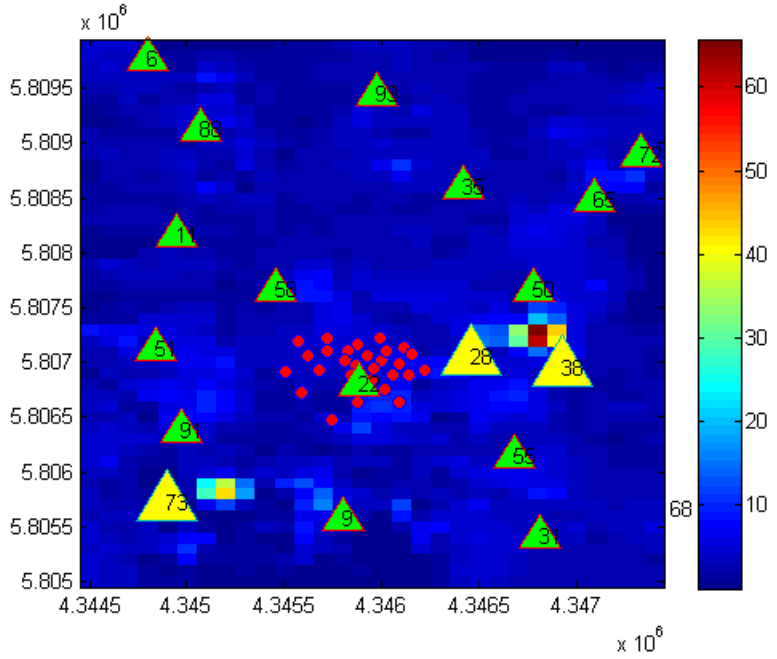

Figure 5.11: Selected BSs by (5.2.5), with capacity scaling 1.09, plotted over the traffic intensity grid. The green triangles represent existing 3-sector sites. The red circles represent micro sites (not selected). The larger yellow triangles are the upgrades selected by the MILP.

way to get any information from the variables. Furthermore, the solution has a (much) lower cost than the solution found in the previous paragraph (namely, no sectorizations selected instead of three). This result means that we have to take a closer look at the dynamic slope scaling procedure.

The dynamic slope scaling procedure attempts to steer the intended binary variables towards binary values by updating the cost function. The update is done in such a way that a non-binary value for a variable leads to a higher cost associated with that variable in the next iteration. This higher cost in the next iteration for all variables that were in the previous iteration non-binary, should steer the LP problem towards a solution with fewer non-binary variables. In [19] we can see that this works quite well for a fixed charge network flow problem. However, we have an extra type of constraint that cannot be modeled in a fixed charge network flow problem: the best server constraint (5.1.7). This constraint has the following form:

$$y_{j_{i_l}} + \sum_{k=l+1}^{L} x_{ij_{i_k}} \leq 1, \qquad l = 1, 2, \ldots, L-1$$

where $(j_{i_1}, j_{i_2}, \ldots, j_{i_L})$ are the base stations that provide the strongest signals to service test point i, in the order strongest to weakest. This constraint gives the assignment of users to base stations more freedom if the variables y_j are not equal to one. In fact, if the variables y_j are as in Figure 5.12 we have a lot

of freedom to assign pixels to base stations that do not provide the strongest signal. This means that the LP problem now balances two objectives:

1. find a solution of lowest value, i.e., distribute the demand as evenly as possible over the base stations,

2. find a solution where all intended binary variables are indeed binary.

These two objectives are clearly not aligned with each other. The result shown in Figure 5.12 shows that the current cost updating scheme tends towards the first objective. We have attempted to steer towards the second objective by adjusting the cost function slightly: we have added 1 to the cost associated with each base station variable. We have done so because the three sector macro sites previously had cost zero and hence the updating scheme did not have any effect (explaining the fast convergence we saw). Note that this changes the problem so we are no longer able to compare it with the results of the MILP. But it is interesting to see if in this case the solution becomes binary for the base station variables. Indeed, the results are shown in Figure 5.13 for the first thirty variables and five iterations. The results show that most variables tend to either approximately zero (everything below 0.001 is regarded as being zero) or one with few exceptions (2 non-binary variables in total). This is indeed what you would expect; most base stations use all of their capacity. We still see that all variables other than the macro 3-sector sites are zero. This might seem strange, since there are only two non-binary base stations selections we expect most traffic to be handled by the best server. So one could expect to see a result similar to the solution of the MILP. Instead we observe no selected upgrades. We took a closer look at the variables that are non-binary and they correspond to base stations 22 and 91. In particular site 22 is a very important site in our network, it is the macro site in the middle of the region where micro sites can be built. A fractional opening of this site means that a large part of the traffic can still be assigned to a base station other than the best server. This explains why we still do not observe any selected upgrades.

The result obtained with this strictly positive cost function seems better at first but we still see that the dynamic slope scaling procedure balances between the two above mentioned objectives. To steer our solution more towards the second objective we have adjusted the cost updating scheme as follows:

$$\left(c_2^k\right)_i = \begin{cases} A \cdot \frac{\left(c_2^0\right)_i}{\left(y_i^{k-1}\right)^3} & \text{if } y_i^{k-1} > 0 \\ M_i^k & \text{if } y_i^{k-1} = 0 \end{cases}$$

where $M_i^k = \max\{\left(c_2^l\right)_i : 0 \leq l \leq k-1\}$. Note the cubic influence of y_i^{k-1}. Tested values of A are $1, 10$ and 10^2. This adjusted cost updating scheme should give an even higher penalty to non-binary variables than the original scheme. The hope is that this will force all base station variables to be binary. The result of this adjustment is however similar to the above result; base station 22 is still opened fractionally in all three cases.

Conclusion:

This example shows that the dynamic slope scaling procedure is not capable of giving a good approximation to the MILP presented in (5.2.5). This result

can only be explained by the best server constraint. After all, without this constraint our problem is in fact a fixed charge network flow problem and as such the results in [19] show that the dynamic slope scaling procedure should be a reasonable approximation. The best server constraint however introduces a conflicting interest. If the intended binary variables are indeed binary we have less freedom in the assignment of pixels to base stations and hence we need more upgrades. However, if the intended binary variables are not binary we have a lot of freedom and we might require less upgrades. The conflict is thus between the intended binary variables being binary and finding a solution of lowest cost.

Since the dynamic slope scaling procedure does not lead to a good approximation of the optimal value of the MILP we will use the exact solution of the MILP in the next chapter.

	1	2	3	4	5
1	0.4798	0.4798	0.4798	0.4798	0.4798
2	0.8898	0.8898	0.8898	0.8898	0.8898
3	0.4798	0.4798	0.4798	0.4798	0.4798
4	0.9066	0.9066	0.9066	0.9066	0.9066
5	0.4798	0.4798	0.4798	0.4798	0.4798
6	0.8632	0.8632	0.8632	0.8632	0.8632
7	0.4798	0.4798	0.4798	0.4798	0.4798
8	0.4798	0.4798	0.4798	0.4798	0.4798
9	0.4798	0.4798	0.4798	0.4798	0.4798
10	0.4798	0.4798	0.4798	0.4798	0.4798
11	0.4798	0.4798	0.4798	0.4798	0.4798
12	0.8149	0.8149	0.8149	0.8149	0.8149
13	0.4798	0.4798	0.4798	0.4798	0.4798
14	0.4798	0.4798	0.4798	0.4798	0.4798
15	0.4798	0.4798	0.4798	0.4798	0.4798
16	0.4798	0.4798	0.4798	0.4798	0.4798
17	0.8228	0.8228	0.8228	0.8228	0.8228
18	0.4798	0.4798	0.4798	0.4798	0.4798
19	0.4798	0.4798	0.4798	0.4798	0.4798
20	0.8909	0.8909	0.8909	0.8909	0.8909
21	0.4798	0.4798	0.4798	0.4798	0.4798
22	0.4798	0.4798	0.4798	0.4798	0.4798
23	0.4798	0.4798	0.4798	0.4798	0.4798
24	0.8314	0.8314	0.8314	0.8314	0.8314
25	0.4798	0.4798	0.4798	0.4798	0.4798
26	0.4798	0.4798	0.4798	0.4798	0.4798
27	0.8064	0.8064	0.8064	0.8064	0.8064
28	0.4798	0.4798	0.4798	0.4798	0.4798
29	0.4798	0.4798	0.4798	0.4798	0.4798
30	0.4798	0.4798	0.4798	0.4798	0.4798

Figure 5.12: Dynamic slope scaling procedure applied to instance 7 of Chapter 6. The first thirty variables (macro base stations) are shown for five iterations.

	1	2	3	4	5
1	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
2	0.7946	0.9727	1.0000	1.0000	1.0000
3	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
4	0.8844	0.9727	1.0000	1.0000	1.0000
5	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
6	0.7907	0.9727	1.0000	1.0000	1.0000
7	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
8	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
9	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
10	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
11	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
12	0.7218	0.7146	0.0130	5.1258e-17	0.0130
13	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
14	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
15	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
16	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
17	0.7621	0.9727	1.0000	1.0000	1.0000
18	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
19	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
20	0.8260	0.9727	1.0000	1.0000	1.0000
21	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
22	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
23	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
24	0.7721	0.9727	1.0000	1.0000	1.0000
25	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
26	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
27	0.7195	3.8558e-10	0.3554	2.8078e-15	5.7589e-10
28	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
29	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12
30	1.2765e-14	4.2548e-15	3.4082e-13	9.7328e-18	2.5037e-12

Figure 5.13: Dynamic slope scaling procedure applied to instance 7 of Chapter 6, with adjusted cost function. The first thirty variables (macro base stations) are shown for five iterations.

Chapter 6

Results

In the previous chapters several methods have been developed to solve the problem posed in Section 1.3. In this chapter we present numerical results obtained using these methods. In particular we investigate the performance of our greedy method with the swap neighborhood versus the greedy method with the 2-step approach. These methods have been thoroughly described in Chapter 4. Another important parameter that we have varied is the choice of initial state. We compare the performance of the above mentioned local search methods using two initial states. (1) the existing network and (2) the network as chosen by the methods of Chapter 5. In a local search method the initial state can be a big influence on the performance and, in particular, the running time. In Chapter 5 we formulated the problem posed in Section 1.3 as a mixed integer linear program. The solution to this mixed integer linear program will form the second initial state.

We will compare the two methods (each with two initial states) by analyzing their performance on a set of problem instances.

The structure of this chapter is as follows. We first give a detailed description of the problem instances. Secondly, we discuss some parameters in the algorithms that had not been chosen yet. Thirdly, we present and compare the numerical results for each of the methods. Finally we will briefly summarize our results.

6.1 Problem instances

We have based our problem instances on one scenario that was made available to us in SONlab. This scenario has a list of potential macro and micro sites. It is important to note that for all problem instances we use the same list. What we do change is the traffic intensity grid. In this section we first describe the existing scenario. Then we explain how we designed our set of problem instances. We also give a detailed description of the problem instances.

The scenario we base our problem instances on covers a 3x5km^2 Hannover region that is also used in the SEMAFOUR project. The area covers both the center of the city as well as some more rural areas. The scenario further consists of a list of potential base stations and an existing network configuration (which is part of that list). The reason we use this scenario is that it was available in

SONlab and we decided to use SONlab because of its great level of detail.

As we have mentioned before, we will create the different instances by varying the traffic map. We therefore present the original traffic intensity map. The traffic intensity map was created by TUBS. The abbreviation TUBS stands for TU Braunschweig (one of the partners in the SEMAFOUR project). The traffic intensity map was made available to us in SONlab in the collection 'busy_hours' with subkey '2013.05.10T16:00'. We will refer to this map as the basic traffic intensity map. We refer the reader to Chapter 3 for more a more detailed description of how this traffic intensity map was created and what it represents. The basic traffic intensity grid contains a few outliers. It is therefore difficult to see the variations clearly. To show the pattern we have, in Figure 6.1, replaced all data points by the minimum of 0.1 and the original data point. The key characteristics of the scenario can be found in Table 6.1. The same holds for the computation of the antenna gain. The application of the antenna gain to the predicted paths is called antenna masking. The outdoor ray-tracing path loss prediction has been discussed in more detail in Chapter 3.

Scenario	LTE 1800
Area Coordinates	E: 4344500 ... 4347500
	N: 5805000 ... 5810000
RATs / Layers	LTE (2 layers: macro + micro)
Existing network	Macro only
Macro cells	LTE 1800
	Transmission power: 46 dBm
Micro cells	LTE 1800
	Transmission power: 30 dBm
Path loss prediction	Pre-calculation of (TUBS outdoor ray-tracing
	+ antenna masking) → masked prediction
Traffic	TUBS traffic intensity map + hotspots

Table 6.1: Scenario specification.

As we can see in Table 6.1 the scenario has potential macro and micro site locations built in, these can be seen in Figure 6.2. The green triangles represent the macro sites, the green circles the micro sites. The **existing network** is formed by the 3-sector macro sites only, so there are no micro sites active. The **upgrade options** we consider are the following:

- A 3-sector macro site can become a 6-sector macro site (6-sectorization).
- A micro site can be activated.

The traffic grid as we have presented it above is the basis for the problem instances we have created. We have created the problem instances using the following procedure. We start from the basic traffic intensity map and add a certain traffic demand using the functions Add_Hotspot, scale_rectangle and scale_rectangle2. The pseudo code of these functions can be found in Appendix C, Functions 1, 2 and 3. The resulting traffic map is then evaluated for two network configurations. First, the existing network (that is: all 3-sector macro sites but no micro sites) and secondly the network with all possible upgrades selected (that is: all 6-sector macro sites and all micro sites active). The

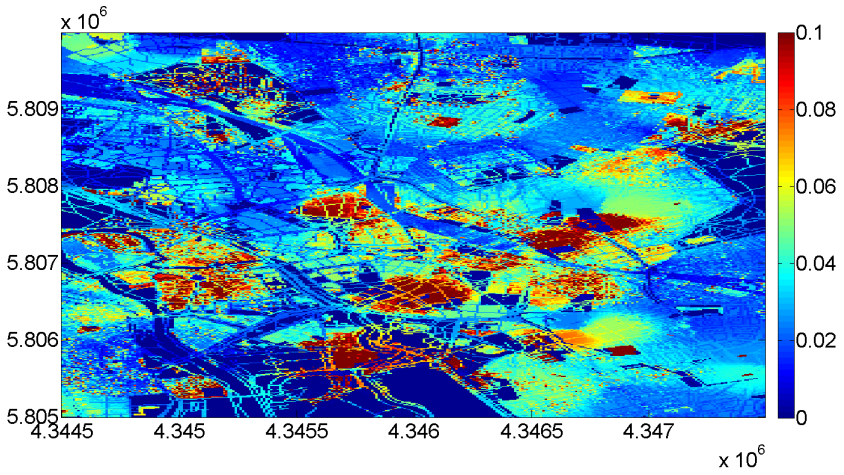

Figure 6.1: The traffic intensity grid capped at 0.1 on the 3x5km^2 Hannover region.

evaluation with the existing network should show a clear problem area. The second evaluation should show no bottleneck. The reason we perform this last evaluation is to ensure that the problem instance can be solved by selecting all upgrades. It is thus a feasibility check for the problem instance.

Remark. In the above feasibility check we assume that adding a site to the network configuration leads to extra capacity in the network. However in practice it might be the case that activating a new site without adjusting the power settings of neighboring base stations might lead to a lower capacity. Calling a problem instance feasible only if the network with 'all upgrades selected' does not show a bottleneck means that we might call certain problem instances infeasible that can, in fact, be solved by a subset of all upgrades. This does not mean that our feasibility check does not check if the instance is feasible, it merely means that our feasibility check is not the best one possible[1].

The specifics of the 19 problem instances that we have defined can be found in Appendix B, Table B.2. As an example we have represented instance 7 graphically in Figure 6.3. In that figure we see the network lay-out as we have seen it earlier together with two circles in which we have added additional traffic (in the form of a Gaussian bell curve, for more details see Appendix B). From the figure it is clear that it is not likely that micro sites will be activated to solve this problem, as the micro sites are far away from the created hotspots. This is not the case for all of the problem instances as can be seen in Appendix B. The problem instances can be divided in two groups; those of which we expect the problem to be solved with the use of micro sites and all others. We expect a problem instance to require the use of micro sites if the hotspots are created within the region where potential micro sites are located and they are small enough in terms of range to be handled by a micro site. Using this categorization

[1]For two feasibility checks A and B we call A a better feasibility check than B if for all problem instances A classifies an instance feasible if B does so.

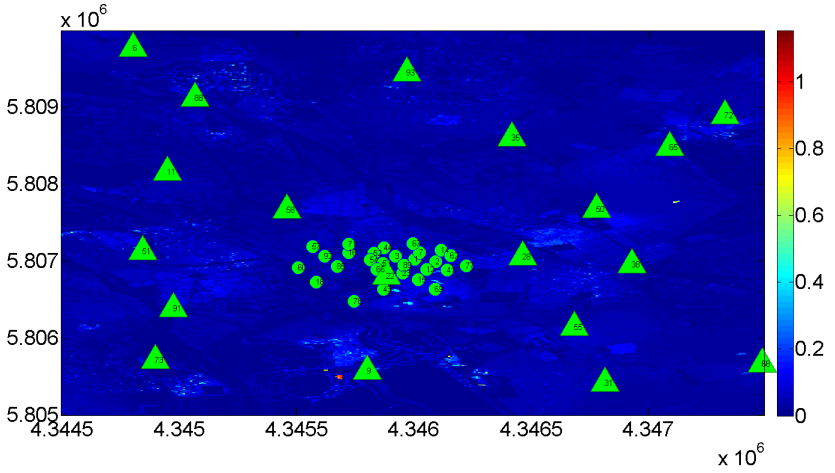

Figure 6.2: The network lay out plotted over the intensity grid on the 3x5km^2 Hannover region. The green triangles represent macro sites, the green circles micro sites.

we can say that the first 7 instances probably do not require the use of micro sites and instances 8 through 19 might require the use of micro sites.

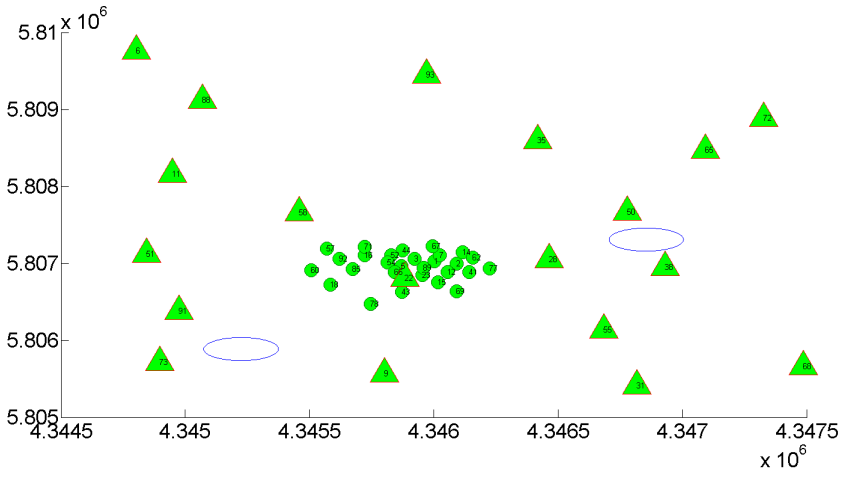

Figure 6.3: The network lay out with a representation of the areas where traffic was added.

6.2 Algorithm parameters

The methods described in Chapter 4 require some parameters to be set. In this section we will give these settings and explain why we chose them in this way.

First of all, there is the traffic filler neighborhood. As we have mentioned in Chapter 4, we need to define two distances. The first distance describes how far a macro site can be from the pixel with the highest overload such that we should still consider it in the neighborhood. The second distance does the same for micro sites. Since our problem zone can be (and usually is) much larger than the single pixel with the highest overload we have chosen these distances rather large. We consider macro sites within 5 km from the pixel with the highest overload and micro sites within 1 km. These distances are larger than the observed 'ranges' for macro (resp. micro) sites, see for instance Chapter 5 Figure 5.8. As we have mentioned, this is because our problem zone has a larger diameter than the single pixel we base our location upon. An interesting point for further research would be to investigate the relation between the size of the problem zone and the distances we use. Which characteristic of the problem zone could best represent the size is, as of yet, unknown. Candidates include the performance cost function, the diameter of the problem zone, and, the largest distance of pixels within the problem zone to the pixel with the highest overload.

Secondly there is the swap neighborhood. In the description in Chapter 4 we mention in the first point a probability to select a type of upgrade that we should remove. The distribution we proposed in that section would come down to the following distribution (using the costs given in Chapter 4):

$$\mathbb{P}(\text{select a 6-sector site}) = \frac{2.3}{2.3+1} \approx 0.7, \quad \mathbb{P}(\text{select a micro site}) = \frac{1}{2.3+1} \approx 0.3.$$

Instead we have used the following distribution:

$$\mathbb{P}(\text{select a 6-sector site}) = 0.6, \qquad \mathbb{P}(\text{select a micro site}) = 0.4.$$

The reason we use this distribution is a historical/practical one. We started testing our methods before the correct prices were known so we used a good guess.

The final setting of the local search methods we need to discuss is the stopping criterion for the local search methods. We have chosen this number by running our greedy method for several instances with a very large number of iterations. The instances we have used are numbers $12, 13$ and 19. We used the data to determine the maximum amount of rejections that was followed by an acceptance. The number we obtained was 12 iterations. Even though not each state has the same size of neighborhood the instance sizes are comparable and as such we can base our stopping criterion on this number. The number was obtained with a small amount of instances and runs. Hence, due to the stochasticity of our methods, we cannot say with certainty that 12 is enough. Therefore we choose a slightly larger bound: 15 iterations.

Remark. The cost function can also be seen as a parameter of the algorithm. In Chapter 4 we have explained that we use estimates of the financial cost as given to us by Nokia [14]. This is however a parameter the network operator should adjust.

6.3 Numerical results

The greedy method with the swap neighborhood and the greedy method with the 2-step approach have both been applied to all 19 problem instances. To test the influence of the initial state we ran the algorithms with two different initial states:

1. the existing network (i.e., only all 3-sector macro sites are active),
2. the network selected by the MILP formulated in Chapter 5.

So in total we could say that we have four methods to test. As we have said, we have applied these methods to all of our problem instances. Because of the stochastic aspects of the methods we ran them more than once: we ran them three times for each instance[2].

We want to compare the methods on three aspects:

1. the solution value of the best state found,
2. the time it takes to find that state,
3. the number of neighbors we accept before we find that state.

The first aspect we will measure per run and average over the runs. That means that we get an averaged best found solution value per instance per method. To get an idea of the spread between runs we will also present the best found solution (value) per instance per method (i.e., the best of all runs). For the sake of completeness we also give the number of macro/micro sites selected (for respectively 6-sectorization or deployement). We will measure the time it takes

[2]We will sometimes see a large difference between the best and the average solution value. This indicates that we should perform more runs. However, since each run is quite time-expensive, we were not able to do so.

to find the best state in the number of iterations needed. Since the computation time per iterations is almost constant, this is a good measure of the running time of an algorithm. The third aspect is mainly of interest when comparing the greedy method with the swap neighborhood to the 2-step approach. The idea is that the 2-step approach will need less accepted neighbors because it has more information to determine the search upon.

6.3.1 The numerical value of the solution of the MILP

The MILP formulated in Chapter 5 has been solved for each of the problem instances. In Table 6.2 we present the solution value per instance, note that the solution value directly translates to the number of 6-sectorizations and micro site activations selected. Since the results showed only 6-sectorizations we omitted a zero column for the activated micro sites. Two instances resulted in an infeasible MILP and for two other instances we observed a solution value of zero. A solution value of zero indicates that the existing network satisfies the capacity bounds. However, since we know that there is a problem zone in the SONlab evaluation, this means that the capacity bounds are too large in these instances (not tight). Therefore, for these four instances the MILP does not provide an initial state other than the existing network. In the tables presented in the next subsection we excluded these instances.

Instance	# 6-sectorizations selected by the MILP	Sites selected
1	Infeasible	
2	0	
3	0	
4	Infeasible	
5	1	73
6	2	28, 50
7	3	28,38 and 73
8	2	28, 58
9	2	28, 58
10	2	28, 58
11	1	58
12	1	58
13	1	58
14	2	28, 58
15	2	28, 58
16	2	28, 58
17	2	28, 58
18	2	28, 58
19	2	28, 58

Table 6.2: The amount of upgrades selected by the MILP per instance.

In later subsections we will see that the amount of upgrades selected by the local search methods is always larger than that of the MILP. This further indicates that the capacity bounds we use are rough estimates at best.

6.3.2 Comparison of the local search methods

In Appendix B, Tables B.4, B.5, B.6 and B.7, we have given the numerical results for respectively the greedy with swap and the 2-step approach. The first two tables show the results of the methods with as initial states the existing network and the last two tables those of the methods with as initial state the solution of the MILP. In this section we will analyze the numerical results. We first briefly comment on whether or not the number of iterations used was sufficient. Then we will compare the local search methods. We have compared the local search methods and use of initial states in the following ways:

1. for each search method we compare its performance per instance using the two different initial states: the existing network or the solution of the MILP

2. we compare the 2-step approach to the greedy method with the swap neighborhood on all instances (using both initial states)

Recall, we expect that the 2-step approach will need less accepted neighbors because it has more information to base the search upon. So in the above mentioned point (2) we are mainly interested in the number of accepted neighbors.

Instances with a lot of traffic demand

In the above mentioned tables we can see that there are several instances for which the average number of iterations required to find the best solution in a run is close to 100. Recall that we force our local search methods to stop after 100 iterations, regardless of whether or not a local optimum is reached. Therefore, if the average number of iterations after which the best solution is found is close to 100 we can conclude that the local search method stopped before reaching a local optimum. For us this is the case for instances $12, 15$ and 17. If we look in Table B.2 at the description of these instances we see that these instances have the largest traffic demand. The three instances are all formed by an increase of the traffic demand in the micro area. Instance 12 also introduces three small hotspots and one larger hotspot (small and large refer to the range). Instances 15 and 17 on the other hand create a narrow, but very high, peak. These three instances are the only instances with these characteristics. We can conclude that for these three instances the maximum number of iterations was too low, for both initial states. We had hoped that the initial state selected by the MILP would speed up convergence and perhaps keep the number of iterations needed below 100. This was not the case. If we look at the third and fourth columns in Tables B.4, B.5, B.6 and B.7, then we see that the best solution requires a lot of upgrades. Table 6.2, however, shows that the MILP only selected one or two upgrades for these instances: a lot less. It is therefore not an unexpected result. At a later point we will discuss the value of using the solution of the MILP as initial state we will come back to this.

Infeasible instances

For instances $4, 5$ and 7 both local search methods are unable to find a feasible solution. But not because the maximum number of iterations was reached.

The local search methods concluded that the problem itself was unfeasible. For instance 4 this claim is correct, we redid the feasibility check and indeed the problem turned out to have a high overload even with all upgrades selected, this can be attributed to a programming error. Instances 5 and 7 also have very high load in the feasibility check, around 0.607. When designing the problem instances we assumed this would be ok, since we also use a certain threshold for the overload. However, the results show that this slight overload on cell level still leads to an unacceptable overload on pixel level. This does not make these instances useless. It is still interesting to see what the MILP did for these instances. We can see that the MILP was feasible for both instance 5 and 7 and we respectively select 1 and 3 sites to upgrade. Although 3 is the largest number of upgrades selected by the MILP, these numbers do not stand out when we compare them to the other instances. From this we can conclude that the solution of the MILP does not provide much insight in how close to infeasibility an instance is. Instance 4 did show an infeasible MILP, but so did instance 1 and that instance did prove to be feasible.

Instances where the initial state provided by the MILP is an improvement

Another set of instances that is of special interest is formed by the instances $6, 14, 16$ and 19. In these instances we see a significant improvement on the number of accepted neighbors required to find the best solution. With the solution of the MILP as initial state we consistently need only two changes. In Figure 6.4 we have displayed this in a bar graph. It should be noted that (average) best found solution value using the initial state obtained from the MILP is, per method, equally good as that found using the existing network as initial state. The figure clearly shows that the number of accepted neighbors

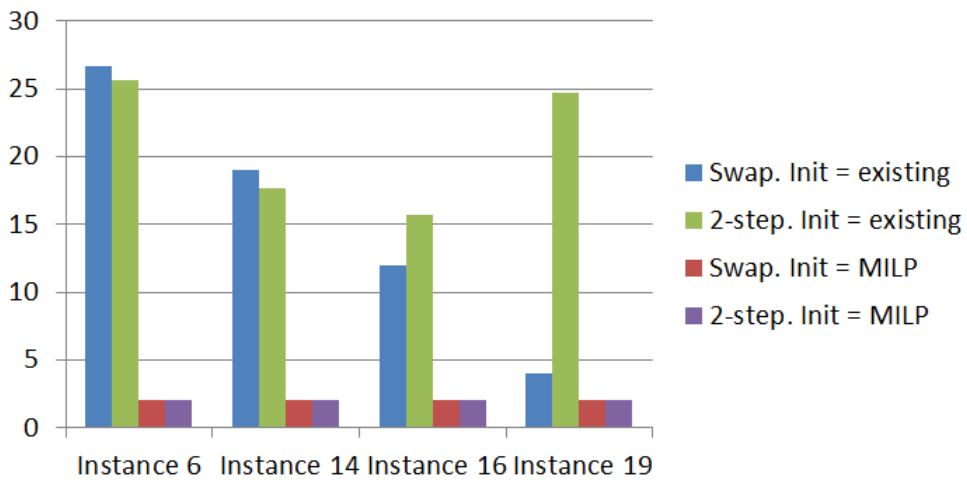

Figure 6.4: The average number of accepted neighbors for each of the four methods on instances $6, 14, 16$ and 19.

reduces substantially when we use the initial state obtained through the MILP.

In the above figure we have only shown the number of accepted neighbors. We could also look at the number of iterations required, but this gives a similar pattern. In Figure 6.5 we have shown both the required number of accepted neighbors as the number of iterations for instance 18. In fact, the only instances

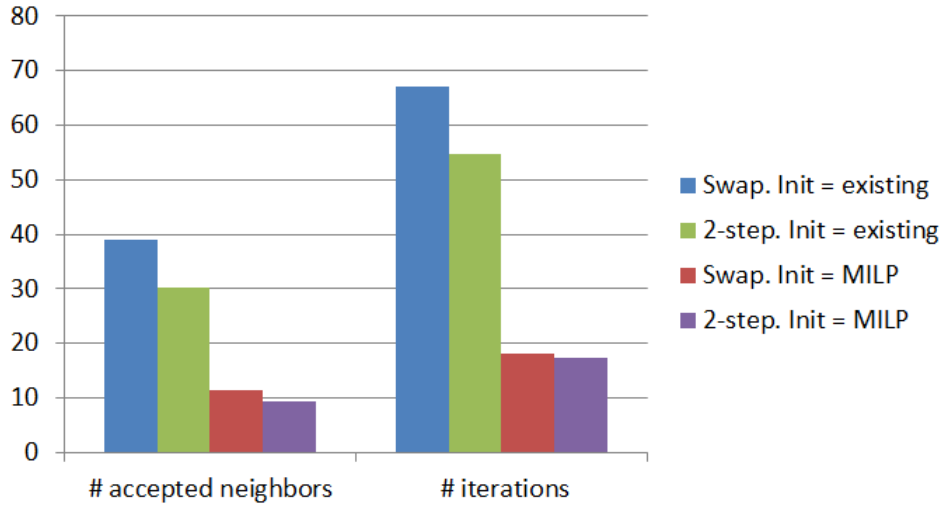

Figure 6.5: The average number of accepted neighbors and iterations for each of the four methods on instance 18.

in which the use of the solution of the MILP did not lead to a time improvement are the ones mentioned in the first two paragraphs (those that either require more than 100 iterations or are infeasible).

The greedy method with the swap neighborhood versus the 2-step approach

The second comparison we would like to make is between the greedy method with the swap neighborhood and the greedy method with the 2-step approach. As we have mentioned earlier, we would expect these methods to behave differently in terms of the number of accepted neighbors (because the 2-step approach bases its search on more information). In terms of best solution value found we do not expect (and do not see) a structural difference. In Figure 6.6 we have displayed the difference in solution value between the two methods for all instances and initial states. On the horizontal axis we have the instances (first with the existing network as initial state), on the vertical axis the difference (swap minus 2-step). We are interest in the distribution among non-zero differences in the categories negative and positive. Therefore we omitted all instances that gave a zero difference and, since the names are not important, we did not label the horizontal axis. To not skew the data we also removed the infeasible instances. The resulting graph shows that in 8 instances the greedy method with the swap neighborhood performed better, in 6 instances the 2-step approach gave a better solution. Note that the ratio is more or less the same for both initial states (this cannot be seen in the graph, but the reader could verify this using the tables in

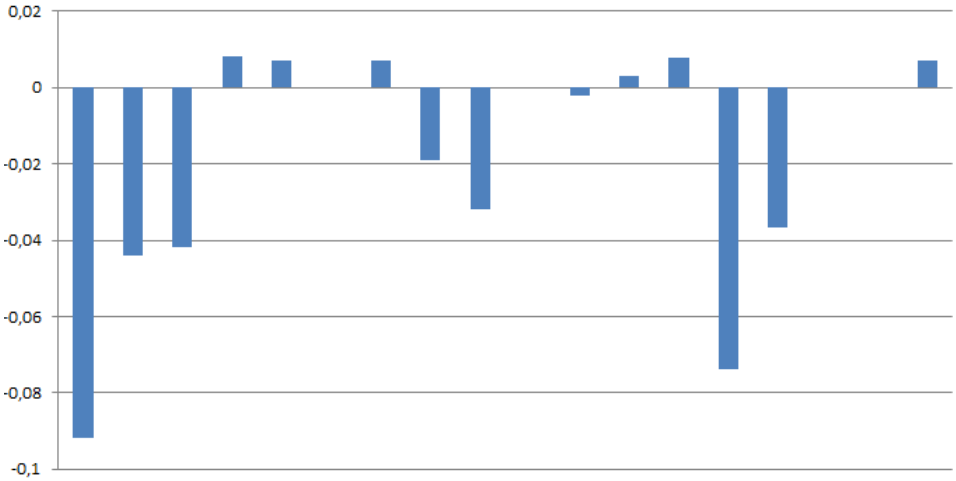

Figure 6.6: The difference in average solution value for the search methods greedy with swap minus the 2-step approach.

Appendix B). We do want to point out that when there is a difference in performance on instance the difference is usually small when the greedy method with swap neighborhood performs worse, while the in the other case the difference seems larger.

From a mathematical point of view, it is not surprising that the best found solution value is more or less the same for both methods. After all, if we are in a feasible state[3] the swap neighborhood is identical to the combined neighborhood of the 2-step approach: namely remove one upgrade and replace it with another. Therefore the local search methods have the same freedom of movement. What is different is the probability distribution with which a neighbor is selected.

In terms of accepted neighbors we expected to see a difference between the greedy method with the swap neighborhood and the 2-step approach. When we look at the difference (again swap minus 2-step), in Figure 6.7, we see however that there is no structural difference. If anything, we could say that the method with swap neighborhood requires slightly less accepted neighbors. However the differences are in general so small that it can also be the result of stochasticity in the methods.

[3]In an infeasible state we cannot use the swap neighborhood. Likewise, in an infeasible state, we will never temporarily allow a step in the 2-step approach.

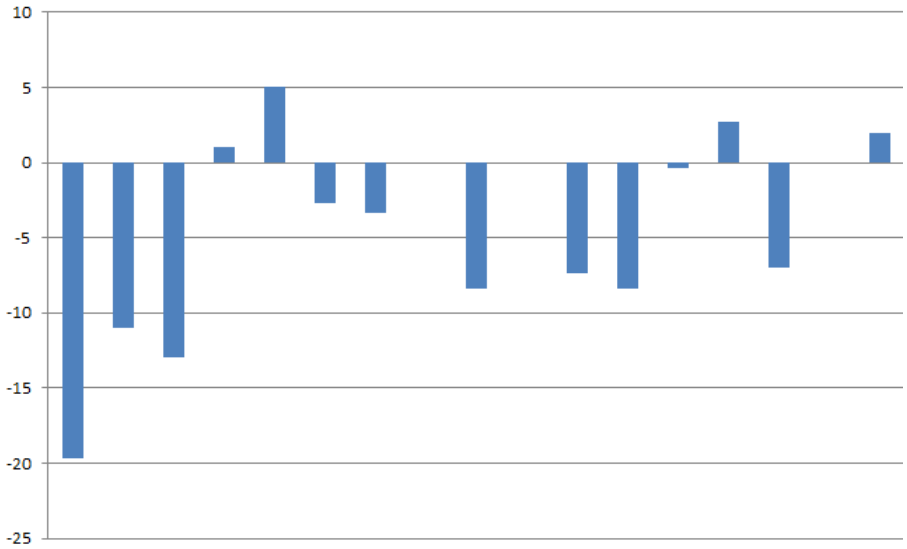

Figure 6.7: The difference in average solution value for the search methods greedy with swap minus the 2-step approach.

6.4 Summary of the numerical results & reflection

In the previous section we have presented the numerical results. We have seen that for both local search methods it is beneficial, for the computation time, to use the initial state obtained from the MILP. In terms of solution value there is no difference between the two initial states (i.e., the existing network versus the solution of the MILP).

Remark. Although in our tests we did not observe a difference in solution value, we did observe a difference in computation time. Therefore, if in a situation the computation time is limited, then it is to be expected that choosing the initial state wisely will also improve the solution value.

We have also compared the two local search methods, i.e., the greedy method with the swap neighborhood and the 2-step approach, but we did not find large differences. If anything, the results show a slightly better performance of the greedy method with the swap neighborhood. But I would recommend further testing before drawing any conclusions. Recall that we expected the 2-step neighborhood to require less accepted neighbors. After all, we designed it to use additional information about the problem zone created by removing an upgrade, in order to search more in the good direction (in this case both literally and figuratively). The numerical results however did not confirm this. This can be caused by two things:

1. Further testing is required on the existing instance database. After all, performing more runs of both methods would mean that the averages converge. These true averages might make the comparison more clear.

2. The instances do not cover all possible problem zone configurations.

Option one does not seem likely, since if it were the case, then we would expect to see at least some indication of it in our limited number of runs. We would like to further clarify why option two might be the case. This is again linked to our expectation of the performance of the 2-step approach. We believe that the current scenario (i.e., the locations of potential sites) does not allow enough freedom to create problems where it is useful to have the extra information about the problem zone. That is, we would like our search algorithm to be in a state as in Figure 6.8. The situation we sketch is the following. One site gets removed (either in the first step of the swap neighborhood or in the first step of the 2-step approach) and its removal causes a high overload further away from the removed site. The swap neighborhood would then select a site to activate that is close to the removed site, while the 2-step approach would (correctly) select one close to the newly formed problem zone. In the current scenario it is for one not possible

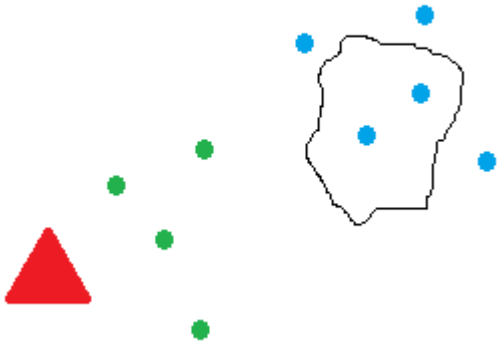

Figure 6.8: Sketch of a situation where the 2-step approach might perform better. The red triangle is a (macro) site that is being removed. This would cause a problem zone indicated by the area of which we showed the boundary in black. The swap neighborhood would select one of the green (micro) sites, the 2-step approach one of the blue (micro) sites.

to completely remove a macro site, we can only undo a 6-sectorization. In itself this would not be a problem: the same situation can also occur when shutting down a micro site (which is possible). However, the potential micro sites are not located close enough to let a situation like in Figure 6.8 occur. The new problem zone would at worst lie around the green micro sites, such that both the swap neighborhood and the 2-step approach would choose from the same set of micro sites (with roughly the same probability distribution).

Chapter 7

Conclusions & Reflection

The previous chapters have been about one question. We will repeat it here, from the problem description:

What is 'the best way' to alter the existing mobile cellular network in order to prevent a predicted future bottleneck?

In Chapters 4 and 5 we introduced the mathematical techniques that we have used: local methods combined with a smart choice of the initial state. In Chapter 6 we presented a comparison between the local search methods, with either the existing network or a smart(er) choice of initial state. The numerical results showed that the two local search methods, the greedy method with the swap neighborhood and the 2-step approach, performed equally well on the test instances. We did see an improvement in terms of computation time when we used the initial state obtained with the methods from Chapter 5. The main question of this thesis can thus be answered in the following way. The 'best way' to alter the existing mobile network can be found using either of the local search methods, as long as the initial state used is that obtained from the MILP presented in Chapter 5.

The best way that we have found in this thesis is not necessarily the best way possible. In fact, we can see several components that can be improved upon. In the next sections we will address two topics that could lead to improved methods. These topics do, however, require further research. In the interest of pointing out further research: in Section 4.5 we have indicated how something called a problem zone can be used to help our local search methods. Also, in Appendix D we present the evaluation tool that was developed during this graduation work. At the end of that appendix we point out areas of improvement for that tool.

7.1 The mixed integer linear program

The mixed integer linear program that we have presented in Chapter 5 uses the base stations as variables. This leads to on one hand a natural formulation of the problem but on the other hand it also creates a problem. In Section 5.4 we have addressed the issue of finding suitable capacity estimates. It turns out that this can best be done on a cell level instead of on a base station level. In

Section 5.4.4 we have also remarked on why we did choose base stations as our variables: it is unclear what the units are in the traffic intensity grid used by SONlab. The method we used is a very rough estimate of the capacity of a base station. It is based on the following: pixels are distributed homogeneously over the cells of a base station, once they are assigned to that base station. This automatically means that theoretical bounds will be hard to give on a base station level. So, to a future researcher we would recommend to try to find a good sense of which units are used and then use the mixed integer linear program on a cell level. The code to write the mixed integer linear program on a cell level is already available, only the right capacity estimates are needed. For the interested reader, we already applied this mixed integer linear program on cell level to several of our instances. We used the capacity estimates found in Section 5.4 method 1. This method did not provide good results, way too much upgrades were selected. In fact, a lot of the instances resulted in infeasible programs, the feasible programs provided too much upgrades compared to the local optima found using the local search methods. This indicates that the averaging we did in method 1 does not provide a good estimate: the estimate is too low. Method 3 could be used as an alternative, provided that the right conversion from Mbps to the units used in SONlab can be determined.

7.2 Multiple possible base stations at one location

So far we have dealt with at most two possible base stations at one location: macro sites can either have 3 sectors or 6. It is however a natural question to ask how the methods presented can be extended to use a list in which an arbitrary (but finite) number of potential base stations can be placed at one location. This is a very relevant question! In such a list we could for example store different rotations of the 3- and 6-sector sites, such that the azimuth becomes a parameter. Another important option is to include several copies of a potential base station with different power settings. Let us discuss our search method from the initialization to the end. The binary and mixed integer linear program formulated in Chapter 5 contain a constraint of the following form:

$$y_j + y_{j'} \leq 1, \qquad \forall (j, j') \in J_{3/6} \tag{7.2.1}$$

This constraint is actually a specific instance of a broader class of constraints. Namely one in which for each location L we have a set of base stations K_L that can be placed on that location with constraint:

$$\sum_{j \in K_L} y_j \leq 1$$

It is clear that (7.2.1) is a specific instance of this broader class. Indeed, we can take the sets K_L to be the pairs $(j, j') \in J_{3/6}$ corresponding to location L.

The local search methods would, however, need to be rewritten (drastically). If we speak about several possible azimuths on one location, then it might be possible to adjust the traffic filler neighborhood in a logical way. There is one azimuth that best corresponds to the problem area. However, it is unclear how the local search methods could deal with different power settings. In [29], on

which we initially based our methods, they do deal with different power settings. We would suggest to extend our neighborhood with something similar to what is done in [29].

Appendix A

Basic elements of a mobile cellular network

We begin by defining some terminology specific to telecommunications networks. We make a distinction between definitions commonly used in the literature and terminology we use at TNO. The concepts in the second category, marked as Informal Definition, could also be defined formally, but we choose not to do so. The goal of the second category is to explain the concepts in as much detail as is needed for this thesis.

A.1 General terminology related to mobile networks

A telecommunications network can be looked at from two different perspectives, the demand side (i.e. the user perspective) and the supply side (i.e. the network operator perspective, the infrastructure). The demand comes from users connected to the network. In literature related to the LTE technology (more on that in Section 1.2) users are referred to as User Equipment, the following definition is commonly used.

Definition A.1.1 (User Equipment (UE))**.** User equipment is any device used by a customer demanding a connection to the network at a specific location and with a certain data rate (in bits/s).

In this project we are not interested in such detailed information, instead we look at area based statistics. Following the notation used in [29] we consider Service Test Points.

Informal definition A.1.2 (Service Test Point (STP))**.** A Service Test Point is a location demanding a connection to the network and a certain data rate (in bits/s). This data rate is the accumulation of the data rates requested by users in a certain region around the location.

The size and shape of the region around the location can vary. As shape we use squares, with as center the above mentioned location. The size of the squares is usually 10 by 10 meters. This will be the smallest size we use, however

to speed up computations we occasionally use a larger size. The concept of an STP is not always described with the words Service Test Point, for example [35] introduces the term *demand node* with the same definition.

The supply side of the network can be summarized as a set of equipment at certain locations providing the connection and data rates requested by the users. The more elaborate discussion of the supply side will run along the following path. First we look at the locations. Secondly we describe the equipment we can place at these locations. Thirdly the terms associated with the connection made between users and the equipment. As a fourth and final topic we discuss some metrics used to evaluate the performance of the network.

As mentioned we start by first define the locations we use and more importantly what options we have at these locations.

Informal definition A.1.3 (Site)**.** A place where a base station can be located.

In general multiple types of base stations can be build at a certain location, i.e. multiple sites could be tied to the same location. However, only one type of base station can actually be build at each location. Note that a list of sites in a network would be a discrete list. This definition raises the question of what a base station is. This brings us to the second point on our path.

Definition A.1.4 (Base station (BS))**.** A base station is a collection of equipment, at one location, which facilitates the communications between user equipment and a network.

In 3G networks base stations are also frequently called node B's, likewise in LTE the abbreviation eNodeB (evolved Node B) is widely used. We will not discuss in detail the collection of equipment mentioned above, the only thing we want to mention is the antennas. Below we give an informal definition of what an antenna means to us.

Informal definition A.1.5 (Antenna)**.** An antenna is a piece of equipment which is able to transmit data in a focused beam and at a certain power level.

In Figure A.1 we can see what it means for an antenna to transmit in a focused beam[1]. The half power beam width, θ_{3dB} in Figure A.1, is a measure for how focused the radiation pattern of the antenna is. The half power beam width is equal to twice the angle between the center of the beam and the angle at which you only receive half the power you could have received at the same distance in the center of the beam. Different antennas have a different half power beam width. Since an antenna is directive it is important to know where it is aimed at. The horizontal angle we call the azimuth, the vertical angle the tilt.

Definition A.1.6 (Azimuth)**.** The angle in the horizontal plane between the center of the beam and some fixed reference direction.

The fixed reference direction in the definition above could for example be north.

[1] A distinction can be made between omni-directional antennas (which transmit to all directions with the same intensity) and directive antennas (which have a radiation pattern as in Figure A.1). The class of antennas we are interested in is the latter so from here on we consider all antennas to be of the directive type.

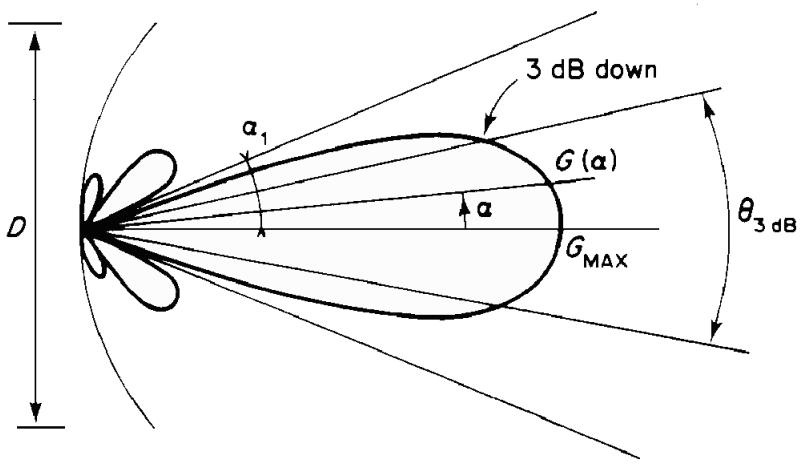

Figure A.1: A model of the radiation pattern of an antenna. The half power beam width is shown with the angle θ_{3dB}.

Definition A.1.7 (Tilt)**.** The angle in the vertical plane between the center of the beam and some fixed reference direction

It is standard to assume the reference direction of tilt to be the horizontal direction.

The radiation pattern we can see in Figure A.1 makes it reasonable to assume that only users which are in a certain direction from the antenna are likely to be linked to it. This brings us to the third point on our path and the term cell or, as it is sometimes called, sector.

Definition A.1.8 (Cell/Sector)**.** The geographical region in which all users (or STPs) are connected to the same antenna array.

Note that a cell does not need to be a connected region. Using this definition it makes sense to speak about a cell belonging to a base station when the antenna array in question belongs to the base station.

Now that we know the definition of a cell we still need to know how the assignment of users to base stations is done. This is based on the best server principle, where the best server of a user is determined as the antenna for which the pilot signal reaching the user is strongest. The pilot signal is a signal which all antenna broadcast at a specific frequency and a certain power setting. The strength with which a signal reaches a user depends on the propagation of radio waves. This is highly influenced by the terrain (buildings, roads, trees, cars, etc.) between the BS and the user. In practice, propagation is therefore something which cannot be determined with certainty without measurements and these measurements are only valid until something or someone moves. This is why path loss models need to be used. In Chapter 3 and Appendix D we discuss two network evaluation tools and the models they use.

As we have mentioned before different types of antennas have a different type of half power beam width. Based on the previous paragraph, it is not hard

to imagine that if the half power beam width is smaller (and the same power setting is used) then the cell will be narrower as well (in this context narrower should be intuitively clear, more precisely we could say that the cell is mostly contained in a smaller cone originating at the antenna). Having more than one cell per base station is very efficient in reducing the costs, however having too many cells means the interference from adjacent antennas will be high. The number of directions used is also the number of sectors of a base station. The **number of sectors of a base station** is thus equal to the number of antenna arrays with a (pairwise) different directionality.

We consider base stations with 1, 3 or 6 sectors. In the case of 3 sectors the directions will differ by 120°, in the case of 6 sectors by 60°. When using a 6-sector base station it is evident that antennas with a smaller half power beam width are preferred. Table A.1 provides some example settings based on the specifications of two Kathrein Scala Division antenna types (the 40°Dual Beam Single Band Panel Antenna for the 6-sector case and the Kathrein 742215 model for the 3-sector case). It is important to note that during the project we

#Sectors	G_m	HPBW_h	HPBW_v
3	18 dB	65°	6.2°
6	20 dB	43°	14.5°

Table A.1: Values used in the antenna gain formula's for 3- and 6-sector sites.

will not vary the antenna type, we will use one antenna type per class of base stations (1-,3- or 6-sectors).

The second parameter determining the cells is the transmission power of the antenna, P^{BS}. The unit we use is decibels (this holds for all future references of power). We will shortly refresh the readers understanding of this unit. The decibel is a dimensionless unit, it says something about the ratio between two units. We say that the ratio R in decibels (dB) between to units I and I_0 is equal to

$$R = 10 \cdot^{10} \log \left(\frac{I}{I_0} \right) [dB]$$

In our case we consider power levels, which are given in terms of Watts. The reference intensity I_0 which is used as a standard in the field is 1 mW.

Conversely when we know the ratio between two intensities in decibels we know the ratio between the intensities by the formula

$$\frac{I}{I_0} = 10^{R/10}.$$

There are two reasons for why we use the unit decibel instead of Watt. The first is that in terms of decibels the power levels become more easily comparable. For example the transmission power is usually around 46 dB whilst the receiver sensitivity of a user is around -90 dB, in terms of Watts this is $10^{4.6}$ vs 10^{-9} mW. The second and perhaps most important reason is that there are various losses between the transmission and reception, by using the logarithmic scale we can use addition instead of multiplication. Since we use it quite often: a loss in intensity of 50% is equal to a loss of about $3,010$ dB, which we usually approximate by 3 dB.

To give an idea of how the power level changes between transmission and reception at the user end we give the following relation:

Informal definition A.1.9 (Received power)**.** The power received at a user, P^r [dB] can be given by the following formula (all variables use the unit decibel):

$$P^r = P^{BS} + G_A - L_A + G_M - L_M - G_h(\phi) - G_v(\theta) - PL - IPL - CL - BL + S$$

where

- P^{BS} is the power usage of the base station (BS)
- G_A, L_A are the antenna gain and loss respectively. They depend on the type of antenna.
- G_M, L_M are the receiver gain and loss respectively for users. Their sum is usually taken as 0 dB [1].
- $G_h(\phi)$ and $G_v(\theta)$ depict the loss due to the radiation pattern not being uniform, depends on the horizontal angle ϕ and the vertical angle θ between user (STP) and BS (approximating formulas exist and will be given in the discussion of my own simulator).
- PL is the path loss.
- IPL is the outdoor to indoor penetration loss, this depends on the material of the walls. Since we only consider outdoor users we can set it to 0 dB.
- CL is the cable loss. The power usage of a BS is measured at ground level, before the actual transmission in the antenna you lose some power in the cables. This typically is around 3 dB.
- BL is the body loss. A typical user holds his/her phone close to their body, which means that part of the signal travels through the body, hence there is a loss here as well. A value of 3 dB is often used.
- S is a stochastic variable for shadowing. $S \sim N(0, \sigma)$ where $\sigma \approx 6$ dB. This variable arises because we have to approximate the path loss. More details about this variable are given in Chapter 3.

As we can see the transmission power of a base station is a parameter, we will distinguish two types of base stations: those with a high transmission power of 46 dB and those with a low transmission power of 30 dB. In terms of sites these two types form respectively the *macro and micro sites*. Intuitively the macro/micro can be understood as the size of the area in which the sites signal is received. There are other differences between macro and micro sites. Base stations at macro sites are usually placed at a high altitude (100m or higher) whilst base stations at micro sites are closer to street level (around 30m). Below we will give an informal definition of macro and micro sites.

Informal definition A.1.10 (Macro sites)**.** A base station located at a high altitude ($>$ 100m), a transmission power of around 46 dB and either 3- or 6-sectors.

Informal definition A.1.11 (Micro sites)**.** A base station located at a low altitude (around 30m), a transmission power of around 30 dB and 1 sector.

The final step along our path is to look at various measures of the performance of a telecommunications network.

A.2 Performance metrics of a mobile network

The performance of a telecommunications network can be measured in various ways (e.g. the load, call blocking rate or user data throughput. The meaning of these terms will become clear at the end of this section). The key performance indicator (KPI) we consider is the load. The load of a network is a vector, each cell has its own load. This load per cell can be seen as the ratio between the physical resources requested by the users connected to that cell and the total available resources of that cell. The total amount of available resources is called the available bandwidth or spectrum. The available spectrum depends on what the operator has bought at the auction, a typical spectrum would be between 1800 and 1810 MHz. Below we give a more thorough description of how the load vector can be computed. We will come to the conclusion that the computation of the load of each cell is a fixed point problem.

Definition A.2.1 (Load)**.** The load of a cell can be given as a fraction between the amount of Herz requested by the users assigned to that cell and the bandwidth, B [Hz], available.

$$\rho = \frac{\sum_{\text{pixels p assigned to the cell}} (\#\text{Hz requested by pixel p})}{B}$$

A pixel is assigned to the cell which provides it with the highest pilot signal (a signal which all BSs broadcast at the same strength).

The number of Herz requested of a base station by a pixel is a complicated number to calculate. It depends strongly on the Signal to Interference plus Noise Ratio (SINR) received at the pixel.

Definition A.2.2 (SINR)**.** Given the received power at an STP by all BSs we can calculate the SINR (Signal to Interference plus Noise Ratio) for each BS at that STP:

$$\text{SINR} = \frac{P^r}{I + N}$$

where P^r is the received signal, I [dB] the interference generated by the surrounding BSs and N [dB] the noise factor.

The noise factor is usually approximated by the following formula:

$$N = 10 \cdot {}^{10}\log(k \cdot B \cdot T) + N_R$$

where k [J/K] is Boltzmann's constant , B [Hz] is the available bandwidth, T [K] the temperature and N_R the noise number of the receiver (around 8 dB). For the temperature we will assume 290K, the available bandwidth can be something in the range of $10 - 20$ MHz (this depends on what the operator bought at the auction, it is part of our scenario description). The interference is the sum over all other cells of their load times the power received from that BS. The reason that we take the load as a factor in this calculation is that, on average, a BS is transmitting at full power for that percentage of time.

The amount of bps a base station can provide to the pixel can be found via Shannon's formula:

$$R_{bps} = 0.6 \cdot B \cdot {}^{2}\log(1 + SINR)$$

where SINR is taken as its linear value, not in dB.

We can then compute the fraction of the bandwith requested by this pixel:

$$\frac{D_{bps}}{R_{bps}}$$

The load of a cell is then the sum over all pixels assigned to that cell of their required fraction of bandwidth.

Now remember that the SINR depends on the load of all other cells. So the load of all cells depends on the load of all cells! This means we are looking at a fixed point problem. This fixed point problem is illustrated in Figure A.2. With an estimate ρ_i of the load of each cells we can find a new estimate of the load ρ_{i+1} by using the old one to estimate the SINR for each pixel and then compute the new load via the above calculations.

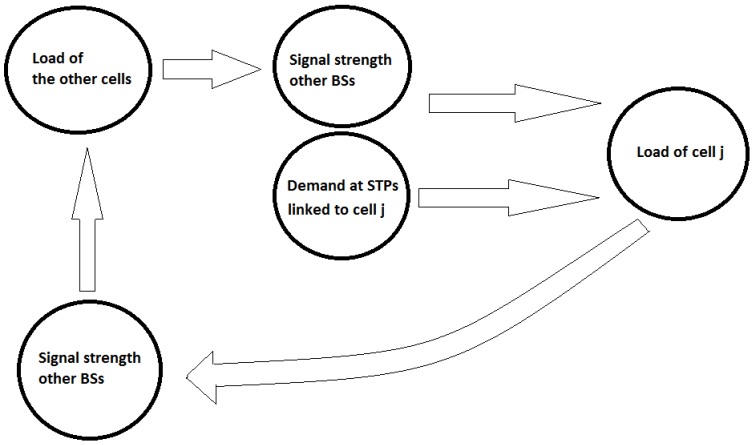

Figure A.2: Illustration of the fixed point problem of calculating the load of cell j.

The desired outcome is that the sequence $(\rho_i)_{i\in\mathbb{N}}$ converges, in practice this is the case.

The above assumes that a user always receives the amount of data it requests. However when the load of a cell approaches one it can happen that there is not enough bandwidth available to provide the required amount of data to the user. In that case either the user data throughput (the data the user actually receives) is less than what it requested. Or the call of that user is blocked by the cell. The probability that a call is blocked by its cell is called the blocking rate. The blocking rate is strongly connected to what we at TNO have called the overload traffic, which is a measure of how much of the traffic intensity is handled by an antenna with a load above the threshold.

Informal definition A.2.3 (Overload traffic)**.** The overload traffic *Overload* is a metric assigning to each pixel the amount of demand handled by a cell with a load above the threshold.

$$\text{Overload\_Traffic}_i = D(x_i) \cdot \sum_{cells j:\rho_j \geq \rho^*} p(x_i, j) \cdot \frac{\rho_j - 0.6}{\rho_j} \tag{A.2.1}$$

where x_i is pixel i, $D(x_i)$ is the demand of pixel i, ρ_j is the load of cell j, ρ^* is the threshold load and $p(x_i, j)$ is the assignment probability of pixel i to cell j.

Appendix B

Tables

	Simulation time step:				
Computation step:	1	2	3	4	5
Get landcover map	1,72	0,00	0,00	0,00	0,00
Pre0001	2,88	0,00	0,00	0,00	0,00
Complete calculation	63,33	36,43	38,25	40,18	36,50
Retrieve traffic	34,60	14,64	14,70	15,25	14,69
Retrieve load	5,69	1,18	1,17	1,23	1,18
get_max_datarate	8,61	8,48	9,49	8,25	8,90
get_ctoi	9,34	8,53	9,20	12,02	8,22
traffic.get_grid	4,90	3,47	3,58	3,32	3,40
Saving load/traffic	0,03	0,03	0,03	0,03	0,03
Calculating bottleneck	0,05	0,00	0,00	0,00	0,00
Connecting to the server	0,04	0,17	0,15	0,13	0,23
Second time connecting to server	0,04	0,00	0,00	0,00	0,00
Third time	0,15	0,00	0,00	0,00	0,00

Table B.1: Running time for 5 simulation steps of the 'busy hour' traffic grid.

Instance number:	Function:	Location:	Peak:	Range:
1	Add_Hotspot	35	1	300
2	Add_Hotspot	89	0.75	250
3	Add_Hotspot	89	1	150
	Add_Hotspot	$(2 + 12 + 41)/3$	0.5	100
4	Add_Hotspot	91	1	300
	Add_Hotspot	$(2 + 12 + 41)/3$	0.5	100
5	Add_Hotspot	$(9 + 73 + 91)/3$	0.5	150
6	Add_Hotspot	$(50 + 38)/2$	0.75	150
7	Add_Hotspot	$(9 + 73 + 91)/3$	0.5	150
	Add_Hotspot	$(50 + 38)/2$	0.75	150
8	scale_rectangle	Micro	1.5	
9	scale_rectangle	Micro	1.35	
	Add_Hotspot	$(43 + 85)/2$	0.4	50
10	scale_rectangle	Micro	1.35	
	Add_Hotspot	$(43 + 85)/2$	0.4	50
	Add_Hotspot	$(1 + 7)/2$	1.5	25
	Add_Hotspot	$(52 + 54)/2$	1.5	25
11	scale_rectangle	Left Micro	2	
12	scale_rectangle	Left Micro	2	
	Add_Hotspot	$(5 + 89)/2$	1.5	35
	Add_Hotspot	$(3 + 52)/2$	1.5	35
	Add_Hotspot	$(15 + 69)/2$	2	25
	Add_Hotspot	$(2 + 12)/2$	0.7	350
13	scale_rectangle2	Micro	0.075	
14	scale_rectangle	Micro	1.5	
	Add_Hotspot	(44+67)/2	1.5	50
15	scale_rectangle	Micro	1.5	
	Add_Hotspot	(85+92)/2	3.5	50
16	scale_rectangle	Micro	1.5	
	Add_Hotspot	(2+41)/2	2.5	50
17	scale_rectangle	Micro	1.5	
	Add_Hotspot	(57+92)/2	4.5	50
18	scale_rectangle	Micro	1.5	
	Add_Hotspot	(44+67)/2	1.5	50
	Add_Hotspot	(7+14)/2	1.5	50
19	scale_rectangle	Micro	1.5	
	Add_Hotspot	(2+62)/2	1.5	50
	Add_Hotspot	(7+14)/2	1.5	50

Table B.2: Problem instances. The locations are mostly given in terms of site coordinates, for example location (50+38)/2 is in the middle of the line between site 50 and site 38. The locations *Micro* and *Left Micro* are defined in Table B.3. The functions are defined in Appendix C.

	Micro	Left Micro
xmin	4345300	4345300
xmax	4346700-10	(4345300 + 4346700)/2 - 10
ymin	5806400	5806400
ymax	5807900-10	5807900-10

Table B.3: Definition of locations Micro and Left Micro.

Instance	Best solution value	# Macro	# Micro	Average solution value	Average # accepted neighbors	Average # iterations
1	0,023	1	0	0,023	2	2
2	0,106	2	6	0,118	28,67	53,33
3	0,102	4	1	0,11	32	60
4	41,41	15	2	41,86	15,67	33,33
5	12,29	8	0	12,78	13	51,67
6	0,069	3	0	0,077	26,67	45,67
7	13,13	12	2	13,13	15	40
8	0,046	2	0	0,046	3	3
9	0,046	2	0	0,046	3	3
10	0,046	2	0	0,046	3	3
11	0,046	2	0	0,046	3	3
12	0,466	12	19	0,473	56,33	96,33
13	0,069	3	0	0,072	20,67	30,33
14	0,069	3	0	0,079	14,33	24,67
15	0,076	2	3	0,145	46,33	80,67
16	0,069	3	0	0,069	7,33	11
17	0,089	3	2	0,264	62,33	96,67
18	0,079	3	1	0,09	23	39
19	0,069	3	0	0,076	16,33	26,33

Table B.4: Greedy method with the swap neighborhood. The initial state used is the existing network. The averages are taken over all 3 runs.

Instance	Best solution value	# Macro	# Micro	Average solution value	Average # accepts	Average # iterations
1	0,092	4	0	0,115	21,67	42,67
2	0,086	2	4	0,162	39,67	70
3	0,132	4	4	0,152	45	76,33
4	41,41	16	2	41,42	19,67	43,67
5	12,29	8	0	12,78	10,33	37,33
6	0,069	3	0	0,069	25,67	52
7	13,13	13	2	22,22	15	30,33
8	0,046	2	0	0,046	3	3
9	0,046	2	0	0,046	3	3
10	0,046	2	0	0,046	3	3
11	0,046	2	0	0,046	3	3
12	0,42	10	19	0,466	51,33	90
13	0,069	3	0	0,072	23,33	34
14	0,069	3	0	0,072	17,67	33
15	0,089	3	2	0,164	46,33	79
16	0,069	3	0	0,101	15,67	24
17	0,076	2	3	0,11	56,67	82,67
18	0,079	3	1	0,092	30,33	54,67
19	0,069	3	0	0,069	24,67	33

Table B.5: Greedy method with the 2-step approach. The initial state used is the existing network. The averages are taken over all 3 runs.

Instance	Best solution value	# Macro	# Micro	Average solution value	Average # accepts	Average # iterations
5	12,29	8	1	12,43	11,67	40,67
6	0,069	3	0	0,069	2	2
7	13,12	12	2	13,29	16,67	56,33
8	0,046	2	0	0,046	3	4
9	0,046	2	0	0,049	3	3,67
10	0,046	2	0	0,054	2,67	4
11	0,046	2	0	0,046	2	2
12	0,433	11	18	0,466	52	96
13	0,069	3	0	0,084	21	31,67
14	0,069	3	0	0,069	2	2
15	0,112	4	2	0,156	34,33	74,33
16	0,069	3	0	0,069	2	2
17	0,112	4	2	0,115	65,33	99
18	0,079	3	1	0,082	11,33	18
19	0,069	3	0	0,069	2	2

Table B.6: Greedy method with the swap neighborhood. The initial state used is the solution to the corresponding MILP. The averages are taken over all 3 runs.

Instance	Best solution value	# Macro	# Micro	Average solution value	Average # steps	Average # iterations
5	12,72	13	0	12,92	13	40,67
6	0,069	3	0	0,069	2	2
7	13,13	13	2	13,16	16,67	41,33
8	0,046	2	0	0,046	3	5,67
9	0,046	2	0	0,046	3	4,33
10	0,046	2	0	0,046	3	5
11	0,046	2	0	0,046	2	2
12	0,506	12	23	0,54	49,33	97,67
13	0,069	3	0	0,069	18,33	28,33
14	0,069	3	0	0,069	2	2
15	0,109	3	4	0,193	41,33	91
16	0,069	3	0	0,069	2	2
17	0,076	2	3	0,203	61	99
18	0,079	3	1	0,08233	9,33	17,33
19	0,069	3	0	0,069	2	2

Table B.7: Greedy method with the 2-step approach. The initial state used is the solution to the corresponding MILP. The averages are taken over all 3 runs.

Appendix C

Functions

Function 1. *Add_Hotspot*

Input:
Peak, range, centerx, centery, grid

*Round centerx and centery to a precision of pixel size (*10 *m)*

Processing:

$newgrid := grid$;

for *all pixels* ***do***
    d:= distance of pixel to (centerx,centery)
    if $d \leq$*range* ***then***
        $newgrid.data(pixel) := newgrid.data(pixel) + peak \cdot \exp(-\frac{d^2}{100 \cdot range})$;
    end if
end for

Output:
newgrid

Function 2. *scale_rectangle*

Input:
scale, xmin, xmax, ymin, ymax, grid

Processing:

newgrid := grid;

for *all pixels* ***do***
    if *pixel lies within the rectangle xmin:xmax by ymin:ymax* ***then***
        $newgrid.data(pixel) := scale \cdot newgrid.data(pixel)$
    end if
end for

Output:
newgrid

Function 3. *scale_rectangle2*

Input:
scale, xmin, xmax, ymin, ymax, grid

Processing:

newgrid := grid;

for *all pixels* ***do***
    if *pixel lies within the rectangle xmin:xmax by ymin:ymax* ***then***
        $newgrid.data(pixel) := scale$
    end if
end for

Output:
newgrid

Appendix D

Another evaluation tool: SANlab

Here we would like to describe the mobile cellular network evaluation tool that was developed during this thesis project: SANlab. This tool was developed with the following goal in mind: to provide a fast evaluation of a network configuration. As such the models used in SANlab often use less detailed information than the ones used in SONlab. We finally did not use SANlab for the following two reasons. One, the computation time of SONlab proved to be acceptable for our local search methods[1]. Two, it is not easy to define a realistic scenario and in SONlab a network configuration and traffic pattern were already available. I want do describe it here for two reasons. One, I learned a lot from developing this tool, so maybe the reader can learn something too. Secondly, as this is my thesis, it should reflect the work that I have done. Developing this tool certainly took some time.

In the description of SANlab we follow the same structure as in Chapter 3. However, since we have already given the general descriptions of the models we use in that chapter, here we will only describe the models we used.

D.1 Network aspects

The traffic distribution is given in the form of a pixel map, just like in SONlab. However, we do know the units. The traffic demand per pixel is given in bits per second. The base stations are modeled exactly as in SONlab.

D.2 Propagation environment

D.2.1 Path loss: The COST 231-Hata model

In SANlab we have implemented the COST 231-Hata path loss model presented in the final report of the COST 231 project, [16]. We use this model since it

[1] For an overview of the computation times required by the various functions of SONlab see Table B.1 in Appendix B. That these times are acceptable follows from the results obtained in Chapter 6.

does not use too detailed information and its validity ranges cover the ranges we intended to use it for. We do not use very detailed information because this is not always readily available and secondly the goal of my own simulator was to be faster than SONlab, which forces us to use a more simplistic model.

The path loss (PL) model can be summarized by the following equations:

$$\begin{aligned} PL &= A + B \cdot \log(d) + C \\ A &= 46.3 + 33.9\log(f_c) - 13.82\log(h_b) - a(h_m) \\ B &= 44.9 - 6.55\log(h_b) \\ a(h_m) &= (1.1\log(f_c) - 0.7)h_m - (1.56\log(f_c) - 0.8) \\ C &= \begin{cases} 0 & \text{in small and medium-sized cities,} \\ 3 & \text{in metropolitan areas.} \end{cases} \end{aligned} \tag{D.2.1}$$

In the above f_c [MHz] is the frequency, h_m [m] the height of the mobile user, h_b [m] the height of the antenna and d [km] is the distance between the user and antenna. The determination of the constant C is not clearly defined, for example what would happen if the antenna is in a metropolitan area but the user in a suburban area? In such a case the area around the user could be leading since that is the area where the radio wave reaches the height range of the user.

This model was tested (in [16]) in realistic environments, the validity ranges that followed from those tests are the following:

- Frequencies between 150 and 2000 MHz.
- The antenna height may be between 30 and 200 meters.
- The user heights may be between 1 and 10 meters.
- Distance between site and user should be between 1 and 20 km.

These validity ranges are almost suited for our needs. The only one we need to violate is the last one. The (horizontal) distance between an antenna and user could be as low as 10 meters. As we can see from the model, distances between zero and one kilometer cause the first logarithm in path loss to be negative. Since this model is very simplistic we choose to solve this problem in a very simple way, we lower bound the path loss by a so-called minimum coupling loss of 70 dB. This should result in a more realistic path loss for smaller distances. The choice for a value of 70 dB was made based on Section 4.5.1 of [1].

From the model in Equation (D.2.1) it should be clear that the COST 231-Hata model does not use detailed information about the environment. The only information about the environment which is taken into account is the distinction between small and medium sized cities and metropolitan areas. This means that building heights, trees, etc. are only taken into account on that level, as an expected average for the area type around the user.

D.2.2 Shadowing

In Chapter 3 we were only able to give a general description of shadowing since we do not have access to the exact models used. In this subsection we describe the models that we have used for the correlation between the normal-distributed

(in dB!) shadowing terms. As we have mentioned in Chapter 3, there are two types of correlation that we have to describe:

1. cross-correlation
2. auto-correlation

We will discuss the models that we have implemented for both.

First the cross-correlation. In [46] various correlation models are considered. The authors of [46] recommend the following model because it captures the properties described in Section 3.3.2. The model they recommend is proven to give a positive semidefinite, psd, covariance matrix[2].

For two antennas and a mobile user with angle ϕ (see Figure 3.2) and distance ratio R ($= 10|\log_{10}(d_1/d_2)|$ in Figure 3.2) the correlation between the two shadowing terms is given by

$$h(\phi, R) = h_\Theta(\phi) h_R(R)$$

where

$$h_\Theta(\phi) = \begin{cases} a - (a-b)\frac{\phi}{\phi_0} & \text{if } \phi \leq \phi_0, \\ b & \text{if } \phi > \phi_0 \end{cases}$$

with $0 \leq b < a \leq 1$ and $0 < \phi_0 \leq 180$. And

$$h_R(R) = \max(0, 1 - R/R_0)$$

with $R_0 > 0$ and usually in the range $[6 \text{ dB}, 20 \text{ dB}]$. They suggest taking $a = 1, b = 0$. The parameters R_0, θ_0 can then be used to fit the model to measurements. For θ_0 a value of 60 or 75 degrees is suggested in [46].

Remark. This means that two antenna from the same base station indeed have correlation equal to one. After all we have $\theta = 0$ so $h_\Theta(\theta) = a = 1$ and $h_R(R) = \max(0, 1 - R/R_0) = \max(0, 1) = 1$.

Secondly the auto-correlation. The model of auto-correlation suggested by Szyszkowicz in [46] is the most widely used one. It is an exponential model where only the absolute distance between two STPs is taken into account:

$$h(d) = e^{-d/d_0}$$

where d is the Euclidean distance between two testpoints and $d_0 > 0$ is a parameter called the decorrelation distance. This model has been shown to always give a psd correlation matrix. Measurement studies, [44], indicate that the spatial 50% decorrelation distance of shadow fading is less than 10 meters for urban micro cells, 120 meters for urban macro cells, and 50 to 400 meters for suburban macro cells.

Implementation. The auto-correlation can be computed without any problems, distances between pairs of test points are known. The auto correlation matrix K_{auto} can then be formed using the model in the previous paragraph.

[2]Recall the definition of the covariance matrix Σ:

$$\Sigma := \sigma(S, S) = E[(S - E[S])(S - E[S])^T] = E[SS^T] - E[S]E[S]^T.$$

It should be clear that a covariance matrix is by definition positive semidefinite. So any model of the cross-correlation should also give a psd covariance matrix.

The cross-correlation is a little more complicated. We need the angle between each pair of antennas for each test point. This angle can be found using the relation

$$v \cdot u = \|v\|\|u\| cos(\theta)$$

where θ is the angle between the two vectors u, v. We also need the ratio $|10 \log_{10}(d_i/d_j)|$ where d_i, d_j are the distances from the test point to the antennas i and j, these are all known. For each test point we then have a cross-correlation matrix $K_{cross,STP}$.

Now the question is how do we combine the two? We construct the joint normal distribution as follows. We first take independent normally distributed variables $X_{i,STP} \sim N(0, \sigma)$ for each test point and each BS. Then we write

$$K_{auto} = C_{auto} C_{auto}^T,$$

this can be done by Cholesky decomposition (since K_{auto} is psd). We assume positive definiteness, so we have that C_{auto} is of the same size as K_{auto}. We can then incorporate the auto-correlation by defining the variables

$$Y_{i,:} = C_{auto} X_{i,:}$$

for each i. The notation $X_{i,:}$ is used to denote the vector consisting of $X_{i,STP}$ for all STPs. Next for each STP we have a cross-correlation matrix $K_{cross,STP}$. Again we assume positive definiteness and write

$$K_{cross,STP} = C_{cross,STP} C_{cross,STP}^T$$

by Cholesky decomposition. Then we define

$$S_{:,STP} = C_{cross,STP} Y_{:,STP}$$

The $S_{i,STP}$ then form our shadowing terms.

During the preprocessing stage a realization of the shadowing terms is taken. This realization is then used to model the inhomogeneity of the environment.

Comments on SONlab and shadowing. We know that SONlab uses assignment probabilities of pixels to cells. Since the only probabilistic aspect of the models used in SONlab is the shadowing we have to assume the assignment probabilities are related to the shadowing terms. Broadcasting a pilot signal from all base stations together and taking into account the path loss estimate and antenna gain would give the mean signal strength at a pixel, this can be added to the shadowing term to find a probability distribution on the signal strength from each antenna to each pixel. The assignment probability to cell j can then be defined as the chance that cell j provides the strongest signal at that pixel.

D.2.3 Antenna gain

We would like to note that using a very detailed radiation pattern (also called: antenna diagram) is would be pointless in SANlab. It would lead to a false sense of accuracy since the inaccuracy of the path loss model is larger than the small variations we see in Figure 3.3.

In SANlab we have used approximation formulas that were presented in the paper of Ericsson, [26]. The approximation formula for the horizontal gain is used both in SONlab and in SANlab, we therefore refer to Chapter 3 for its formula. The antenna gain in the vertical direction, $G_v(\theta)$ [dB], is given by the following formula:

$$G_v(\theta) = \max(-12\left(\frac{\theta}{HPBW_v}\right)^2, SLL_v)$$

where SLL_v [dB] is the Side Lobe Level and $HPBW_v$ is the half power beam width in the vertical direction. θ is in radians, it depicts the angle in the vertical plane between the horizontal plane and the user, minus the tilt.

Table D.1 provides some example values based on the specifications of two Kathrein Scala Division antenna types (the 40°Dual Beam Single Band Panel Antenna for the 6-sector case and the Kathrein 742215 model for the 3-sector case). These values are used in SANlab.

#Sectors	G_m	HPBW_h	FBR_h	HPBW_v	SLL_v
3	18 dB	65°	30 dB	6.2°	-18 dB
6	20 dB	43°	30 dB	14.5°	-18 dB

Table D.1: Values used in the antenna gain formula's for 3- and 6-sector sites.

Figure 3.4 compares the proposed model to the radiation pattern of the Kathrein 742215 model.

The antenna gain in the vertical direction is clearly different from the one that is used in SONlab. An analysis of the difference between the two different vertical gain models does not lie within the scope of this thesis, for the reader who is interested in doing such an analysis [26] and [12] form a good starting point, these papers describe the respective models.

D.3 Traffic handling

In Chapter 3 we have described the cell assignment process in SONlab. The cell assignment process in SANlab is very similar. The only difference is that we do not use the full shadowing distribution. Instead, for one scenario, we take a realization of the shadowing and use that as if it is the true correction on the path loss that was required. This allows us to deterministically assign pixels to cells. The load computation is based on the same fixed point problem. However, we solve it in the iterative way described in Appendix A.2.

As an illustration of SANlab we would like to present Figures D.1 and D.2. To create Figure D.1 we have imported the scenario that we have also used in Chapter 6, with SONlab. With that scenario we made a signal strength map. Note that this map is created without shadowing. In Figure D.2 we did take into account shadowing. This time for a fictional scenario with only one 3-sector base station. We took a very small half power beam width to clearly show the directivity of the antennas.

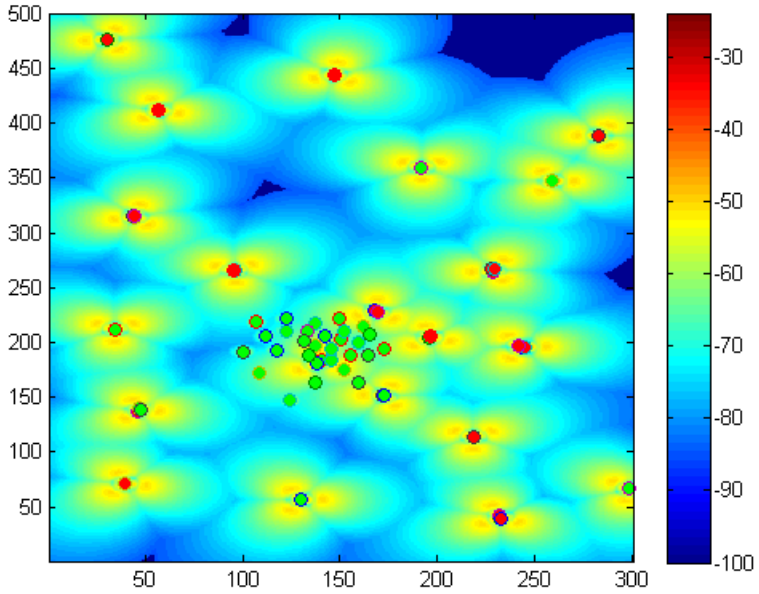

Figure D.1: The signal strength map of the SEMAFOUR scenario used in Chapter 6, as calculated with SANlab. Without shadowing. There is no difference between the red and green circles other than the color.

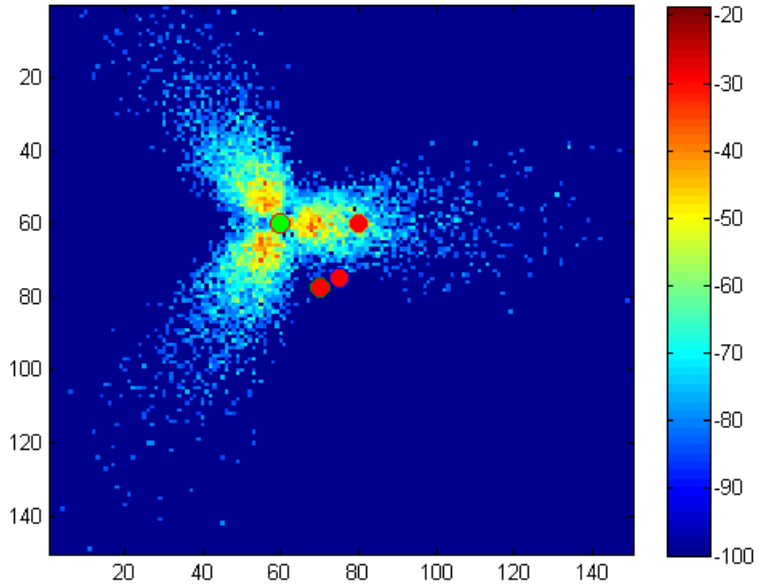

Figure D.2: The signal strength map of a scenario with one active 3-sector site.

D.4 Concluding remarks

We can compare the evaluation tool described in this appendix to the one described in Chapter 3. We can see that the main difference is in the path loss model used. The one in SONlab uses much more detailed information. Also, we do not have access to specific information about the shadowing models used in SONlab. But since the shadowing is a correction on errors made in the path loss model it stands to reason that there will be some differences in that aspect as well.

D.4.1 Further advice to a future user of SANlab

Both SONlab and SANlab offer the opportunity to give a large list of potential base stations and perform a certain preprocessing. This preprocessing can be seen as calculating all aspects of the propagation environment. Doing this in a separate stage beforehand is very useful when you want to evaluate a certain scenario for more than one network configuration (as we do in this thesis). The reason that this works is that for all the propagation models we can take a subset of the base stations and still satisfy the model. For path loss and antenna gain it is clear that this works. To see that it also works for shadowing we note that a principal submatrix[3] of a covariance matrix is the covariance matrix of the remaining subset of variables.

The path loss model we used might not be the best fit for your needs. Please confirm that it is indeed valid for the ranges you are interested in. If it does not, then [23] might be a good place to look for improvements.

This evaluation tool was developed by someone who is not a telecommunications expert. Although I have tried to use only referenced values it would be wise to check if they agree with your intentions. In the cell assignment code you will find a receiver sensitivity (with reference), these values are quite recent so they might not agree with standards set by for example the 3GPP [1].

[3] A principal submatrix of a square matrix A is a square matrix B obtained by removing a set of columns and the same set of rows from the matrix A.

Bibliography

[1] 3GPP Specification 36.942. www.3gpp.org/dynareport/36942.htm.

[2] About the SEMAFOUR project. http://www.fp7-semafour.eu/en/about-semafour/.

[3] atesio. www.atesio.com.

[4] Cisco radiation patterns. http://www.cisco.com/c/en/us/products/collateral/wireless/aironet-antennas-accessories/prod_white_paper0900aecd806a1a3e.html.

[5] CPLEX LP format. lpsolve.sourceforge.net/5.5/CPLEX-format.htm.

[6] GNU Linear Programming Kit. www.gnu.org/software/glpk/#downloading.

[7] Gurobi. www.gurobi.com.

[8] NEOS Server, SCIP Solver. www.neos-server.org/neos/solvers/milp:scip/CPLEX.html.

[9] Public deliverables of the SEMAFOUR project. http://fp7-semafour.eu/en/public-deliverables/.

[10] A. Molina, G.E. Athanasiadou, A.R. Nix. The Automatic Location of Base-Stations for Optimised Cellular Coverage: A new combinatorial approach. *IEEE Vehicular Technology Conference*, 1:606–610, 1999.

[11] Tobias Achterberg. Scip: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009. mpc.zib.de/index.php/MPC/article/view/4.

[12] Z. Altman, B. Begasse, C. Dale, A. Karwowski, J. Wiart, Man-Fai Wong, and L. Gattoufi. Efficient models for base station antennas for human exposure assessment. *Electromagnetic Compatibility, IEEE Transactions on*, 44(4):588–592, Nov 2002.

[13] Carsten Lund, Mihalis Yannakakis. On the Hardness of Approximating Minimization Problems. *Journal of the Association for Computing Machinery*, 41(5):960–981, 1994.

[14] C. Coletti. *Heterogeneous Deployment Analysis for Cost-effective Mobile Network Evolution: An LTE Operator Case Study : PhD Thesis.* Radio Access Technology Section, Aalborg University.

[15] C.Y. Lee, H.G. Kang. Cell Planning with Capacity Expansion in Mobile Communcations: A Tabu Search Approach. *IEEE Transactions on Vehicular Technology*, 49(5):1678–1691, 2000.

[16] E. Damosso, L.M. Correia, Information Market European Commission. DGX III "Telecommunications, and Exploitation of Research.". *COST Action 231: Digital Mobile Radio Towards Future Generation Systems : Final Report*. EUR (Series). European Commission, 1999.

[17] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, Catherine Schevon. Optimization by Simulated Annealing: An experimental evaluation; Part 1, Graph Partitioning. *Operations Research Society of America*, (6):865–892, 1989.

[18] Dennis M. Rose et al. D2.4: Definition of Reference Scenarios, Modelling Assumptions and Methodologies, June 2013. Non-public report of the SEMAFOUR project.

[19] Dukwon Kim, Panos M. Pardalos. A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. *Operations Research Lettters*, 24:195–203, 1998.

[20] E. Amaldi, A. Capone, F. Malucelli, C. Mannino. Optimization problems and models for planning cellular networks. *Handbook of Optimization in Telecommunication*, pages 879–901, 2006.

[21] E. Amaldi, P. Belotti, A. Capone, F. Malucelli. Optimizing base station location and configuration in UMTS networks. *IEEE*, 2011.

[22] Emile H. L. Aarts, Jan H. M. Korst, Peter J. M. van Laarhoven. *Local Search in Combinatorial Optimization*. John Wiley & Sons Ltd.

[23] V. Erceg, L.J. Greenstein, S.Y. Tjandra, S.R. Parkoff, A. Gupta, B. Kulic, A.A. Julius, and R. Bianchi. An empirically based path loss model for wireless channels in suburban environments. *Selected Areas in Communications, IEEE Journal on*, 17(7):1205–1211, Jul 1999.

[24] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533–549, 1987.

[25] Michel Gendreau. An introduction to tabu search. In Fred Glover and GaryA. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research and Management Science*, pages 37–54. Springer US, 2003.

[26] F. Gunnarsson, M.N. Johansson, A. Furuskar, M. Lundevall, A. Simonsson, C. Tidestav, and M. Blomgren. Downtilted base station antennas - a simulation model proposal and impact on hspa and lte performance. In *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, pages 1–5, Sept 2008.

[27] Bruce Hajek. Cooling schedules for optimal annealing. *Mathematics of operations research*, 13(2):311–329, 1988.

[28] H. Holma and A. Toskala. *LTE for UMTS - OFDMA and SC-FDMA Based Radio Access*. Wiley, 2009.

[29] Stephen Hurley. Planning Effective Cellular Mobile Radio Networks. *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY*, 51(2):243–253, 2002.

[30] I. Siomina, A. Furuskar, G. Fodor. A mathematical framework for statistical QoS and capacity studies in OFDM networks. In *IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications*.

[31] Iana Siomina, Di Yuan. Analysis of Cell Load Coupling for LTE Network Planning and Optimization. *IEEE Transactions on Wireless Communications*, 2012.

[32] J. Yang, M. E. Aydin, J. Zhang, C. Maple. UMTS base station location planning: a mathematical model and heuristic optimisation algorithms. *Communications, IET*, 1(5):1007–1014, 2007.

[33] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York.*, pages 85–103, 1972.

[34] Kimmo Hiltunen. *The Performance of Dense and Heterogeneuous LTE Network Deployments within an Urban Environment*. PhD thesis, Aalto University School of Electrical Engineering.

[35] K.Tutschku. Demand-based Radio Network Planning of Cellular Mobile Communication Systems. Research Report Series 177, University of Würzburg, Institute of Computer Science.

[36] L. Hu, I.Z. Kovács, P. Mogensen, O. Klein and W. Störmer. Optimal New Site Deployment Algorithm for Heterogeneous Cellular Networks. *IEEE*, 2011.

[37] Larry Raisanen. A permutation-coded evolutionary strategy for multiobjective GSM network planning. *Journal of Heuristics*, 14(1):1–21, 2008.

[38] M. St-Hilaire, S. Chamerland, S. Pierre. A tabu search algorithm for the global planning problem of third generation mobile networks. *Computers & Electrical Engineering*, 34(6):470–487, 2008.

[39] P. Mogensen, Wei Na, I.Z. Kovacs, F. Frederiksen, A. Pokhariyal, K.I. Pedersen, T. Kolding, K. Hugl, and M. Kuusela. Lte capacity compared to the shannon bound. In *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, pages 1234–1238, April 2007.

[40] R. Sridharan. The capacitated plant location problem. *European Journal of Operations Research*, 87:203–213, 1995.

[41] Roger M. Whitaker, Larry Raisanen, Steve Hurley. A Model for Conflict Resolution between Coverage and Cost in Cellular Wireless Networks. *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, 2004.

[42] Roger M. Whitaker, Steve Hurley. Evolution of Planning for Wireless Communication Systems. *Proceedings of the 36th Hawaii International Conference on System Sciences*, 2003.

[43] Rudolf Mathar, Thomas Niessen. Optimum Positioning of base stations for cellular radio networks. *Wireless Networks*, 6:421–428, 2000.

[44] J. Salo, L. Vuokko, H.M. El-Sallabi, and P. Vainikainen. An additive model as a physical basis for shadow fading. *Vehicular Technology, IEEE Transactions on*, 56(1):13–26, Jan 2007.

[45] Scott Kirkpatrick. Optimization by Simulated Annealing: Quantitative Studies. *Journal of Statistical Physics*, 34(5/6):975–986, 1984.

[46] S.S. Szyszkowicz, H. Yanikomeroglu, and J.S. Thompson. On the feasibility of wireless shadowing correlation models. *Vehicular Technology, IEEE Transactions on*, 59(9):4222–4236, Nov 2010.

Index