



Budapest University of Technology and Economics
Department of Telecommunications and Media Informatics

Methods for Planning of Next Generation Mobile and Data Networks

János Harmatos

*High Speed Networks Laboratory
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics*

Ph.D. Dissertation

Advisor:

Dr. Gyula Sallai

*Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics*

Budapest, Hungary
2003



Budapesti Műszaki és Gazdaságtudományi Egyetem
Távközlési és Médiainformatikai Tanszék

Eljárások Következő Generációs Mobil és Adat Hálózatok Tervezésére

Harmatos János

*Nagysebességű Hálózatok Laboratórium
Távközlési és Médiainformatikai Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem*

Ph. D. disszertáció

Tudományos vezető:

Dr. Sallai Gyula

*Távközlési és Médiainformatikai Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem*

Budapest
2003

Contents

1	Cost-based Planning of UMTS Terrestrial Radio Access Networks (UTRAN)	1
1.1	Introduction	1
1.2	The UTRAN architecture	2
1.3	Problem Statement	3
1.4	The Network Model	4
1.5	The Proposed Planning Method	6
1.6	The RNC localization algorithm (RLA)	7
1.6.1	Genetic algorithm bases	7
1.6.2	The optimization method	8
1.7	The RBS tree construction algorithm (TCA)	10
1.7.1	The Greedy tree construction	11
1.7.2	The optimization process	12
1.7.3	Improved Simulated Annealing	14
1.7.4	The local improvement process	14
1.8	Method for Planning UMTS Networks (MPUN)	16
1.9	Performance Analysis	17
1.9.1	Comparison of different planning methods of UMTS networks	17
1.9.2	Examination of the RNC Localization Algorithm (RLA) . .	18
1.9.3	Examination of the RBS Tree Construction Algorithm (TCA)	19
1.10	Conclusions	21
2	Cost-based Planning of UMTS Core (Backbone) Networks	22
2.1	Introduction	22
2.2	Architecture of the UMTS Core Network	23
2.3	Problem Statement	24
2.4	The Solution	26
2.4.1	Solving Phase 1	27
2.4.2	Solving Phase 2	28
2.5	Post-Processing: Further Improvements on the Solution	31
2.5.1	Capacity decreasing method	32
2.5.2	Residual capacity based re-routing	32

2.5.3	Link capacity extension	33
2.5.4	Link deleting	33
2.6	Application of the methods to solve general network planning problems	33
2.7	Performance Analysis	34
2.7.1	Lower bound calculation for the Phase 1 method	35
2.7.2	Comparison of the classical and improved Simulated Allocation based algorithm	35
2.7.3	Comparison of the two second phase methods	36
2.7.4	Testing the effectiveness of the PostProcess method	37
2.8	Conclusions	37
3	Optimization of OSPF Administrative Weights	39
3.1	Introduction	39
3.2	Problem Statement	40
3.3	MIP Formulation	42
3.4	Outline of the Simple Weight Adjustment (SWA) Algorithm	43
3.5	Outline of the Weight Optimizer (WO) Method	46
3.5.1	Properties of the WO method	46
3.5.2	Structure of the WO method	47
3.5.3	Set_link_weight procedure	48
3.5.4	Load_balancing_#1 procedure	49
3.5.5	Methods in Normal State	51
3.5.6	MAX-MIN_decreasing procedure	51
3.5.7	Variance_decreasing procedure	51
3.5.8	Load_balancing_#2 procedure	52
3.5.9	Framework of the WO algorithm	52
3.6	Performance Analysis	54
3.6.1	Test environment	54
3.6.2	Comparison test	54
3.6.3	Effectiveness of NS state optimization of WO algorithm	57
3.6.4	Issues on scalability of WO	59
3.7	Extension of the WO algorithm	60
3.8	Conclusions	60
4	Dynamic Routing and Wavelength Assignment in Multifiber WDM Networks	62
4.1	Introduction	62
4.2	The Network Model	65
4.3	The Proposed Wavelength Selection and Routing Methods	66
4.3.1	Assumptions for the Proposed Algorithm and Network Environment	66
4.3.2	The Proposed Weight Functions	67

4.3.3	Reference (Existing) Wavelength Selection Strategies	68
4.3.4	The Proposed Wavelength Selection Strategy	70
4.3.5	Issues on fairness between demands	71
4.3.6	Path length examination	72
4.3.7	Pseudo-Code of the Algorithm Frame	72
4.4	Performance Analysis	74
4.4.1	Simulation Environment	74
4.4.2	Simulations on Blocking Probability	75
4.4.3	Issues on Fairness Criteria	81
4.5	Application of the WS&WR algorithm in WDM network configuration	82
4.6	Conclusions	85

Acknowledgement

I would like to express my great appreciation to my advisor Dr. Gyula Sallai.

I would like to thank Dr. Miklós Boda, head of Research and Development Unit at Ericsson and Dr. Hans Eriksson, head of Traffic Analysis and Network Performance Laboratory at Ericsson for their continuous support and encouragement. I would also like to thank Dr. Tamás Henk, head of High Speed Networks Laboratory for his valuable comments.

I would like to thank Dr. Áron Szentesi for his valuable support and encouragement and his useful ideas and advices.

I would like to specially thank to all the colleagues I have worked together with at Ericsson Traffic Laboratory and High Speed Networks Laboratory, especially to Alpár Jüttner, István Gódor, Dr. Gábor Magyar, Zsuzsa Weiner, Dr. Péter Laborczi, Tamás Dékány, Balázs Szviatovszki, Attila Szlovensák, Dániel Orincsay, Balázs Józsa, Dr. Tibor Cinkler and András Veres.

I would like to thank to Piotr Gajowniczek, Andrzej Myslek, Stanislaw Kozdrowski and Michal Piòro for the fruitful cooperation and their help in Thesis 2 and Thesis 3.

I would especially like to thank to my family for their continuous help and encouragement.

I would like to thank to all my friends, who have supported and helped me to walk on my way, especially to Péter Baranyai, Ágnes Gödör, István Gödör, Bálint Háda, István Maricza, István Rimányi, György Suhai, Mónika Szászik, Mónika Simovits, Árpád Szlávik, and Sándor Molnár.

Finally, I would like to dedicate this dissertation to my sweetheart *Eszter Sólyom*.

List of abbreviations

ATM Asynchronous Transfer Mode

GSM Global System for Mobile Telecommunication

ECMP Equal-Cost MultiPath

ILP Integer Linear Programming

IP Internet Protocol

LER Label Edge Router

LSR Label Switching Router

LP Linear Programming

MGw Media Gateway

MPLS Multi-Protocol Label Switching

MP λ S Multi-Protocol Lambda Switching

OSPF Open Shortest Path First

PSTN Public Switched Telephone Network

QoS Quality of Service

RBS Radio Base Station

RNC Radio Network Controller

UMTS Universal Mobile Telecommunication System

UTRAN UMTS Terrestrial Radio Access Network

WDM Wavelength Division Multiplexing

WWW World Wide Web

Chapter 1

Cost-based Planning of UMTS Terrestrial Radio Access Networks (UTRAN)

1.1 Introduction

Nowadays, there is a significant change in the field of wireless technology. Apart from the continuous and enormous growth in the number of subscribers of GSM wireless technology, UMTS (Universal Mobile Telecommunications System) is getting more and more important, because the 3rd generation of mobile networks will be built upon this technology. UMTS will incorporate a multitude of services, with special emphasis on data transfer, and will play an important role in future telecommunications, its penetration is therefore expected to be very high. Because of this and the increasing volume of data communication, the access network of the UMTS will be a rather complex and high-capacity ATM and later an all-IP based system.

UMTS networks will be at least as significant in the future as GSM systems today. Considering the scale of current mobile networks, the importance of good planning methods seems obvious for ensuring the cost-efficient development of UMTS Terrestrial Radio Access Networks (UTRAN). Consequently, algorithmic optimization is essential for the cost-effective and economical planning of these large and complex transport networks; the quality of the solution will determine the long-term performance and service quality parameters of UMTS, which are really critical issues in the competitive mobile market. Therefore, I developed a novel network model and an optimization method conforming to the special, new properties of the UMTS network, taking into account the new constraints for the topology of the network and the strongly non-linear cost functions, which make the optimization problem very difficult.

The cost of the UMTS Terrestrial Radio Access Network is composed of two parts: the cost of equipment in the nodes and the cost of the transmission network. However, while the first part is very much determined by several factors (subscriber distribution, required coverage, radio propagation), we have much more freedom to define the structure of the transmission network. Perhaps the most important goal (besides ensuring the required capacity and quality of service) is to minimize the total cost of the transmission network. There are, however, constraints that limit the range of feasible solutions. These limitations basically stem from the technological characteristics of the equipment used, as well as topological limits on network distances and node degrees. Another kind of limitation can be the presence of an existing transport system that has to be re-used for UMTS.

1.2 The UTRAN architecture

The terrestrial access part of the UMTS network basically consists of two types of network elements called RNC (*Radio Network Controller*) and RBS (*Radio Base Station*) as depicted in Figure 1.1.

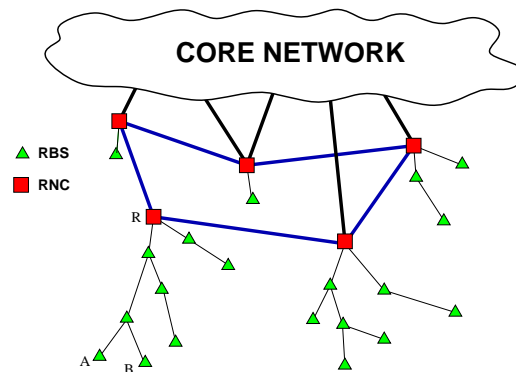


Figure 1.1: UMTS network

The task of the RNC is to manage the radio channels of RBSs connected to it, and it also concentrates the traffic flows of connections and trunks them to the upper level core network. The RBS handles the radio channels and forwards the traffic of lower level RBSs towards their dedicated RNC. In the first phase of introducing UMTS networks the transmission system is built on ATM technology, but the ultimate goal is the all-IP based network. Over the physical ATM/IP topology each RBS is communicating directly with its home RNC using ATM PVCs (Permanent Virtual Connections) or IP packet forwarding, so the logical topology of the UMTS access networks is a set of star subnetworks. On physical level the RBSs are connected to their home RNC either directly or through some other

RBSs in a cascaded way. This implies that the physical level network topology is a set of trees. In each tree there is an RNC as root node and given number of RBSs.

We are given a constraint on how many RBSs can be connected to an RNC, and on how many RBSs can be connected to another one. The reason of this constraint is that the number of ports of RNCs and RBSs are limited in the basic configuration and additional ports means significant cost increasing of the equipment. In the rest of the chapter these type of constraints is called *degree constraints*. There is also a constraint on the depth of the RBS subtree, which means how many other RBSs can be placed between any RBS and the RNC, in other words how many RBSs can be cascaded in the network. The reason of this constraint is that the delay must be fulfilled a pre-defined value, which strictly limits the number of RBSs, which can be cascaded. Further on, this constraint is called *cascading constraint*. The *level* value of an RBS means how many link hops are placed between itself and the RNC in case of the current network topology. In [1] there is further information about the UMTS system.

1.3 Problem Statement

The design problems that appear in UTRANs can be classified into two main groups:

Task 1.: we have to optimize the number and location of RNCs in the network

Task 2.: we have to plan the physical transmission system that connects the RBSs to their corresponding RNCs.

It is assumed that the number and locations of the RBSs and their generated traffic volumes are given as an input for the planning algorithm (these data are based on the results of radio network planning). Another important assumption we make is that we are given a set of geographical sites which can be optional places of RNCs – note that this way we transfer the originally continuous-domain RNC location problem into a discrete optimization problem. We have to take into account the aforementioned constraints and limitations determined by the equipment capabilities (node degree and tree depth constraints). In most of the cases it is also a basic requirement to handle the case if some network elements (RNCs or transmission links) are pre-defined and during the optimization process they cannot be changed or moved (this is required when we want to plan the cost-optimal extension of an existing network).

My goal was to develop a planning method fulfilling all the mentioned requirements. We could not find existing solution in the literature covering every required aspect, although similar problems have already been investigated, for example in the case of GSM networks [9]. This kind of optimization is often referred to as a facility location problem, see for example [2] for a good overview.

While several simple algorithms exist for finding unconstrained minimum-cost spanning trees (see for example Borůvka’s algorithm in [3]), constrained spanning tree problems are typically \mathcal{NP} -hard (including our version as well). In [4] there is a detailed study about the complexity of these kind of problems, and in [5] it is proved that the construction of multi-constrained spanning tree is already an \mathcal{NP} -hard task in itself. In [6], [7] the authors propose solutions for similar problem, but only in case of *one* constraint. However, so far we could not find any solution for the joint existence for the two types of constraints (i. e. degree and depth). In [C1] my colleague propose a two-layer algorithm which solve the two above tasks in an integrated way. I found that if I solve the RNC localization and the tree-building tasks separately using different algorithms, better solution is obtained. This task decomposition results some running time increasing, but because I solved a green-field, preliminary network design task, the quality of the solution is more important metric of the performance of the algorithm than the running time.

Considering the \mathcal{NP} -hard feature of the problem, it is impossible to obtain an exact solution in reasonable running time, so I have to use heuristic approach. The above mentioned requirements and conditions cause that using an efficient heuristic method is not enough in itself, we also need to adjust it to the current special conditions of the optimization problem.

1.4 The Network Model

In this section a novel network model is presented for the UTRAN networks from the viewpoint of network planning.

Let $\mathcal{S} = \{a_0, a_1, a_2, \dots, a_k, \dots, a_N\}$ denote the set of the geographical points (site), where an RBS, or an RBS and an RNC is placed, and N means the number of sites (in the model this equals to the number of RBSs). Two subset of \mathcal{S} is defined for the predefined and possible location of RNCs, namely, R_{pre} and R_{pos} .

Let

$$v_i = \begin{cases} 1 & \text{if RNC is installed into site } i, i \in (R_{pre} \cup R_{pos}) \\ 0 & \text{if site } i \text{ does not contain RNC} \end{cases}$$

Let E denote the set of possible transmission links, and let

$$z_{ij} = \begin{cases} 1 & \text{if there exist a link between } i \in \mathcal{S} \text{ and } j \in \mathcal{S} \\ 0 & \text{if there is no link between } i \in \mathcal{S} \text{ and } j \in \mathcal{S} \end{cases}$$

We use the following notations for the constraints:

- *Cascading constraint* denoted by L
- *RBS degree constraint* denoted by D_{RBS}
- *RNC degree constraint* denoted by D_{RNC}

In our model the total cost of the network consists of the cost of RNC and RBS devices and the cost of the links. We use such kind of general cost description [8], which is able to handle the different cost components of mobile access networks.

The link cost between site i and j is denoted by $Cost_{link}(ij)$ has a capacity independent and a capacity dependent part, and our cost functions are able to handle both wired (leased-line, fiber, coax) and wireless (microwave) interconnections. The capacity independent part of the transmission link cost consists of two parts. One part is proportional to the distance of the sites, while other part is the cost of the repeaters required between site i and j . The capacity dependent part of the link cost is typically represented by an increasing step-like or piece-like function. Our model is able to handle both approaches. The total cost of a link is calculated by the sum of the above components.

The cost of the j th RNC and k th RBS devices are denoted by $Cost_{RNC_j}$ and $Cost_{RBS_k}$. RBS cost consist of four types of subcosts, namely:

- Cost of the number of ports is a step-like function and depends on the number of other RBSs connected to the current one.
- Capacity dependent cost of port is also step-like type and describe the cost of the required capacity of the current port.
- Equipment cost represents the installation and investment cost of required subdevices, namely processors, boards, etc. This cost is calculated as a linear combination of the sum of traffic value passed through the current device and the number of used physical ports.
- Cost of the infrastructure (for example cost of the building in which the RBS or RNC devices are placed can be consider in this cost factor).

In case of RNC we have an additional cost of the required management devices.

The total cost of the whole access network is calculated as follows:

$$Cost_{total} = \sum_{i \in (R_{pre} \cup R_{pos})} Cost_{RNC_i} \cdot v_i + \sum_{k \in S} Cost_{RBS_k} + \sum_{ij \in E} Cost_{link}(ij) \cdot z_{ij} \quad (1.1)$$

Let l_j and d_j denote the current level and degree value of RBS j , let d_{RNC_i} denote the current degree value of RNC i . Let N_{RNC}^{min} denote the minimum number of RNCs to be installed into the network, let N_{RNC} denote the current number of RNCs installed into the network. Finally, let Y_k denote the number of RBSs on $level_k$, $k \in \{1 \dots L\}$.

The purpose is to minimize $Cost_{total}$ subject to the following constraints:

$$\begin{aligned}
l_j &\leq L && \text{for all RBS } j, \\
d_j &\leq D_{RBS} && \text{for all RBS } j, \\
d_{RNC_i} &\leq D_{RNC} && \text{for RNC } i, \\
\sum_{i \in (R_{pre} \cup R_{pos})} v_i &= N_{RNC} && \text{for RNC } i \\
\sum_{k=0}^L Y_k &= \sum_{i,j \in S} z_{ij} && \text{for all site } i, j \\
\sum_{i \in S} z_{ij} &= 1 && \text{for all site } i, j \\
\frac{N}{D_{RNC}} &\leq N_{RNC}^{min} &&
\end{aligned} \tag{1.2}$$

1.5 The Proposed Planning Method

As we could see, the state space of the original task is enormous, therefore it does not seem feasible to create an optimization method that can obtain a reasonable solution for the original problem directly. Therefore, it is better to divide the original task into two subtasks that results significant decreasing of complexity of the problem. The most obvious classification of the original task is the following:

- Task 1.: Finding the optimal number and placement of Radio Network Controllers. A part of this task is to determine the home RNC of each RBS, by other words this is the RBS clustering task.
- Task 2.: Constructing the constrained-tree topology subnetworks of Radio Base Stations that connects them to each others and the RNCs.

The related existing methods handle the two above tasks in a common way, and use integrated, layered algorithms like proposed in [9] and [C1]. The advantage of the integration of the two tasks into one algorithm is the fast operation, but it is obvious that we cannot obtain near-optimal results. Another way if we solve the two tasks in a separate way, using two different algorithms. Although, here we have to pay the price of running time increasing, but the obtained solution can be much better than in the previous case.

In the following Sections I will propose two new algorithms for solving the above mentioned tasks. The short descriptions of the methods can be seen below:

- RNC Localization Algorithm (RLA) [for solving Task 1]: The task solved here is the determination of the RNC locations and the clustering of the RBSs (to assign a home RNC for each RBS). I applied genetic algorithm to solve the problem. Instead of $Cost_{total}$ here a modified goal function is used, namely: $Cost_{mod_total} = \sum_{i \in (R_{pre} \cup R_{pos})} Cost_{RNC_i} \cdot v_i + Cost_{RBS_subtree}$, where $Cost_{RBS_subtree}$ is calculated according to the description presented in Section 1.6.2. (THESIS 1.1)

- RBS Tree Construction Algorithm (TCA) [for solving Task 2]: Here the task is to design a cost optimal multi-constrained tree topology in each clusters that are determined by the previous algorithm. I used an improved simulated annealing method here. This method also contains a post-process, which modify the network topology to obtain better final solution using elementary steps. Our goal here is to minimize the original $Cost_{total}$ function. (THESIS 1.2)

Beside the detailed presentation of the above algorithms I evaluate their results and compare them to existing methods.

1.6 The RNC localization algorithm (RLA)

Here a facility location type problem should be solved, where the input is the set of sites, and the goal is to make near optimal number of clusters from them. I used a genetic algorithm based method to solve this problem, therefore at first I give a brief overview about most important properties of this kind of optimization technique, then the task specific characteristics of the genetic method is outlined.

1.6.1 Genetic algorithm bases

The genetic algorithm is a wide-known general adaptive heuristic method, which may be used to solve search and optimization problems. It is based on the genetic processes of biological organism, namely over many generations, natural populations evolve according to the principles of natural selection (survival of the fittests). Modeling this process, genetic algorithm is able to "evolve" solutions to real world problems, if they have been suitably encoded. A simple genetic algorithm has the following operators: reproduction, crossover and mutation. The algorithm has the following features, which are very useful in case of our optimization problem:

- Effectiveness when the shape of the objective function is completely unknown ($Cost_{total}$ is typically such a function!).
- Difficult problems can somehow be handled, though efficiency is impaired.
- The typical paradigm of genetic algorithm is well matched to the optimization problem like our one.
- The implementation of the algorithm is simple.

Consequently, genetic algorithm can be an efficient tool in our hand to solve complicated optimization tasks, but to get near-optimal results we have to conform the following conditions:

- To find an appropriate modeling of the optimization task to be solved with genes is very difficult, but we have to solve it in the best way, because it determines directly the quality of the final solution.
- The operators (e.g. crossover, mutation) do the proper modifications (which means that crossover of entities results a really fitter one).
- The fitness function is suitable.

If the problem specific version of the genetic algorithm is conform to the above conditions, the quality of the solution will be very good. For more details about the theory of the genetic algorithm we refer to [12].

1.6.2 The optimization method

In this Section the problem specific features of the general genetic algorithm are outlined.

In our approach an entity represent the number and location of RNCs in the network. According to this, an entity is an array of integer values x_i , $i \in R_{pos}$. If in the i th position of the vector $x_i = 1$ then the i th site contains an RNC, while if $x_i = 0$ there is no RNC in the site.

The number and position of RNCs change using the genetic operators described above, such as reproduction, crossover and mutation. The algorithm executes the following steps during the optimization:

Initial state generation In my adaptation an entity is a vector x , and the length of the vector is equal to the possible RNC positions. The number of the population can be adjusted by the user, the minimum number of RNC N_{RNC}^{min} is calculated, then in each entity $\left(N_{RNC}^{min} - R_{pre} + \text{Random}\left(0, \frac{N_{RNC}^{min} - R_{pre}}{2}\right)\right)$ pieces of RNCs is placed into randomly chosen locations.

Step 1. Recombination. I used the following method here: always the two best entities are selected from the previous population (as parents) and the new population will be created from them. By other words this is the so-called elitism.

Step 2. Crossover. Crossover operator is used to create the entities of the new population from the selected parents in Step 1. In a simple case the new entities are built up from the gene of the randomly selected parents, gene by gene. Our previous tests proved that this crossover operator is not effective enough, therefore I elaborated a more complex crossover strategy. In the proposed crossover operation the algorithm uses a special entity called *best*. The i th position of this entity contains information about how many times an RNC was installed in the given position during the optimization. The *best* individual is useful, because it gives us information about the past, thus it stores the advantageous and disadvantageous locations of the RNCs.

In my adaptation the crossover operator compares the two parents, and in case of those position in which the values are different the *best* entity is used to decide that the new entity will inherit which value, according to the following process. We have a probability $P_{inherit_the_better}$ (adjusted by the user) and a random generated number $0 \leq L \leq 1$. If we denote the current step by j and number in the i th position of *best* entity by c then the decision happens in the following way:

- If $L \leq P_{inherit_the_better}$ and if $c > (j - c)$ then in the new entity $x_i = 1$, else $x_i = 0$.
- If $L > P_{inherit_the_better}$ and if $c > (j - c)$ then in the new entity $x_i = 0$, else $x_i = 1$.

Using extended simulations I have shown that this technique gives 2-5% better results than the classical crossover technique.

Step 3. Mutation. In this algorithm we have three ways to apply the mutation operator, namely:

- Simple mutation: Random alternation of the value at a random generated position i of an entity; $x_i = 1 - x_i$. This kind of mutation helps to adjust the near-optimal number of RNCs in the network within few iteration steps.
- Double mutation: In this case two positions are searched in a random selected entity, where $x_i = 1$ in the one position, while $x_k = 0$ in the other position, then both values are altered. This seems to be a very useful operation, because when we have already have the (near)-optimal number of RNCs we do not need to change the number of RNCs, but we have to try as many possible distribution of RNCs as possible to examine quite great state space.
- Strong mutation: In this case an entity is selected randomly and in some positions $x_i = 1 - x_i$. The number of altered positions is also generated randomly and it is between the $[N_{RNC}^{min}, \frac{R_{pos}}{2}]$ interval. This kind of mutation helps to move out the optimization from a local minimum point.

The ratio between the mutations can be set in two ways: In one respect, the first two types are used alternately in each step of the iteration for a random selected entity, while the third one is applied once about in every 10 steps. On the other hand, the probability of mutations are adjusted adaptively:

- In the beginning each mutation has the similar probability.
- If a mutation resulted an entity with lower cost then the probability of mutation will be increased by 2%, and the probability of the two other mutation types are decreased by 1%.

Using simulations I have shown that in case of smaller networks and simpler cost structures, the fixed probabilities are preferable, but in case of larger networks or complex cost functions the adaptive probability setting results better solution.

Step 4. Cost calculation for each entities. It is required to determine the goodness of the current RNC locations, and for the cost calculation we should connect the RBS to their home RNC. In this algorithm three ways are possible to build the whole access network:

- Version 1: connect each RBS directly to the closest RNC
- Version 2: apply a simple greedy method (presented in Section 1.7.1).
- Version 3: apply the proposed TCA (details in Section 1.7) for building up the RBS tree.

Step 5. Continue the optimization at Step 1. until a prescribed number of iteration is reached. It is possible to use an adaptive stopping criteria as well. In this case the optimization is stopped when during a pre-defined step limit $step_{lim}$ the network cost was not reduced. $step_{lim}$ is set adaptively during the optimization in the following way:

$$step_{lim} = \frac{\sqrt{3} * 10}{\sqrt[3]{step_{curr_imp}}} * (step_{curr_imp} - step_{last_imp})$$

,when $step_{curr_imp}$ is the current step where improvement happened, while $step_{last_imp}$ such the last step, when a better solution was reached.

The simulations show that application of the adaptive stopping criteria can reduce the running time by about 20-25% comparing the classical genetic algorithm without performance degradation.

1.7 The RBS tree construction algorithm (TCA)

This process starts out from the output of previous process and builds up the final transmission network between the RBSs and RNCs. The input is the locations of the RNCs and the clusters of RBSs. Then this methods is able to plan a cost-optimal transmission network for the groups of RBSs that connect to their corresponding RNC. I have to note that this method have to be run cluster by cluster to obtain the whole transmission network.

The method is based on the *simulated annealing* heuristic and a modified version of a greedy tree-construction algorithm and a local improvement method (post-processing). The simulated annealing based method is used to optimize the RBS distribution on different levels, while the greedy algorithm builds up the complete

network to calculate the cost of a network state. The local improvement method is used to achieve more optimal final result.

Simulated annealing is a widely used optimization method, known to be able to find a solution close to the global optimum even in cases of large state spaces (see [10, 11] for a general overview). The method works in an analogous fashion to the physical annealing of solids to attain minimum internal energy states. The basic idea is to generate a path through the state space of the optimization, from one state to another nearby state, leading ultimately to the optimum. In generating this path, states are chosen from the locality of the preceding state by a probabilistic function of the improvement gained by this move. So, steps are not strictly required to produce improved results, but each step has a certain probability of leading to improvement. At the start all states are equally likely, but as the algorithm progresses, the tolerance for states worse than the current one decreases, eventually to the point where only improvements are accepted. In this way the algorithm can attain the optimal solution without becoming trapped in a local optimum point. The advantages of this heuristic method are general usability, easy adaptation to a particular application, easy implementation and relatively short running time. Unfortunately, it has several disadvantages, namely there are no good criteria to stop the optimization, and we have no knowledge about the relation between the found local optimum and the global one. We refer the reader to [10] for more details on the principles of simulated annealing.

The state space of the optimization can be calculated from the possible distribution of RBSs among the levels and the possible positions of transmission links in cases of different level structures. The number of possible RBS distributions for first level can be calculated by the following equation:

$$A = \binom{N}{Y_k} \quad (1.3)$$

The number of possible positions of links is calculated in the following way:

$$B = \prod_{l=0}^{L-1} Y_l \cdot Y_{l+1} \quad (1.4)$$

The state space is $A \cdot B$, which is the required step number to find the optimal solution in the worst case.

1.7.1 The Greedy tree construction

The following algorithm is based on Prim's well-known minimal spanning tree algorithm, but we have to consider the capacity dependent cost function and the constraints. The operation of the greedy algorithm is as follows.

During the algorithm the set of RBSs are divided two disjoint sets B and U , namely the RBSs which are already connected to the RNC are in B and the others are in U . At the beginning of the algorithm set B is empty.

In each step, for every RBS $u \in U$ the algorithm determines the $b_u \in B$ such that u can be connected to b_u without violating the constraints and the cost of connection is minimal. Then it chooses the RBS $u_{min} \in U$ with minimal cost of the connection and connects it to the corresponding element in B , places it to B , and repeats this step until set U becomes empty.

To perform these steps efficiently, we maintain a database of the actual best candidate for connection, its cost and its nearest site for all element of U .

If the level of a newly connected RBS is less than L and its nearest site has less than D connected RBSs, then if it is closer to some RBSs which are in set U than their neighbour, then we update its data.

If some RBS reaches its degree constraint, then we have to update the data structure of all RBSs in U for which its neighbour is this site or is in this connecting tree, accordingly.

It is important to note that although using this algorithm the constrained tree could be built very fast, the final cost of the network will much higher than the optimal value, because of the greedy principle.

1.7.2 The optimization process

In our approach we use an adaptive version of simulated annealing presented in section 1.7.3, where a state is the current number and location of RBSs on $level_1$. The exact description of the optimization process is the following:

Initial state generation We can give a lower bound for the required number of RBSs to be on $level_1$ to avoid the bottleneck property of the greedy method. The lower bound is calculated in the following way:

$$Y_{1-LB} = \left\lceil \frac{N}{\sum_{l=1}^L D_{RBS}^{l-1}} \right\rceil \quad (1.5)$$

As initial state we choose Y_{1-LB} number RBSs, which are closest to the RNC to be on $level_1$.

Then the algorithm starts the following iteration process:

Let Y_1^i mean the current number of RBSs on $level_1$ in the i th iteration.

Step 1. Select one of the next alternates with probabilities will be given in section 1.7.3:

- Select an RBS randomly and assign it to $level_1$ with probability P_{add}^i ; $Y_1^{i+1} = Y_1^i + 1$.

- Remove an RBS selected randomly from $level_1$ with probability P_{remove}^i if $Y_1^i > Y_{1_LB}$; $Y_1^{i+1} = Y_1^i - 1$.
- Swap an RBS on $level_1$ and another one from another $level$ with probability P_{move}^i randomly; $Y_1^{i+1} = Y_1^i$.

Step 2. In each step of simulated annealing we have to compute the network cost to decide whether the changes in network topology are acceptable or not. For computing the cost for the current $level_1$ RBSs, we use the greedy method to build up the whole network, with the following modified starting state. Set B will contains the RBSs are on $level_1$, while set U will contains the rest of them.

Step 3. Calculate the cost difference between the new and the original network. On the basis of the following stochastic acceptance criteria the simulated annealing method decides to accept or refuse the new network structure.

The network modification will be accepted with probability

$$P_{accept} = \min\left\{1, \exp\left(-\frac{Cost_{new} - Cost_{curr}}{T}\right)\right\}, \quad (1.6)$$

where $Cost_{new}$ and $Cost_{curr}$ are the total cost of the network after and before the modification and T is the so-called temperature which decreases exponentially during the algorithm.

If the cost of the new state is lower than the cost of the current state, the new state is always accepted. On the other hand, when the cost of the new state is higher than the cost of the current state, the acceptance of the new state depends on a stochastic criterion, as defined above. At the beginning of the optimization the probability of the acceptance of a higher-cost new state is close to 1, while later this probability decreases significantly.

The process continues from *Step 1.* until RBS distribution optimization has finished.

Final Step Network will be optimized by local improvement method (more details see Section 1.7.4).

Extension of the basic algorithm: It is possible to repeat the above optimization process on all levels. In this case the algorithm starts to optimize $level_2$ and, using same method, continues up to $level_{L-1}$. When we are optimizing $level_k$ the RBSs on $level_m$ ($m \in \{0 \dots k-1\}$) have been already fixed and their interconnections has been already determined. So, the lower bound for the number of RBSs on $level_k$ can be calculated in the following way:

$$Y_{k_LB} = \left\lceil \frac{N - \sum_{m=0}^{k-1} Y_m}{\sum_{l=k}^L D_{RBS}^{l-1}} \right\rceil \quad (1.7)$$

My experience was that the optimization on $level_1$ is the critical point of the process, much better result cannot be obtained by optimization on the rest of levels. Furthermore, the local-improvement post-process works more effectively if only the first level RBSs are optimized and the rest of the network is built up by the simple greedy algorithm.

1.7.3 Improved Simulated Annealing

In the original form of simulated annealing, the neighbour states are selected randomly, with the same probability. This results in a wide search of the whole state space, but the algorithm is not “motivated” in any way to avoid the “bad paths” towards local optimum points, which can increase the running time significantly. Therefore, I have improved the classical simulated annealing with the ability of adaptivity. In our method the value of probability P_{add} , P_{remove} and P_{move} in the current iteration depends on the difference of $Cost_{new}$ and $Cost_{curr}$ in the previous iteration. In the following let $\Delta C = Cost_{new} - Cost_{curr}$.

If $\Delta C > \frac{Cost_{curr}}{10}$ then the cost of the network has increased significantly, so it is not good to continue the optimization in this way. Therefore, the algorithm decrease the probability of the last selection by 5 percent. For example, if the last selection has added a new RBS to $level_1$ with P_{add}^i , in the next iteration $P_{add}^{i+1} = P_{add}^i - P_{add}^i \cdot 0.05$. If $\Delta C < 0$, it means that the network cost decreased, caused by the last selection, therefore the algorithm increase the probability of the last selection and decrease the two other ones by the 5 percent. It means that there is a *feedback* in the simulated annealing, so it is able to react to the effect of the last selection. This method can significantly increase the convergence speed resulting shorter running time.

Another change in the improved version of simulated annealing takes into account the topological characteristics of the current network. It means that in case of selecting RBS swapping, the possible neighbour RBSs are located within a given region around the current one. Namely, if a_m is on $level_1$, the algorithm can only swap it for RBS a_n , if $dist(a_m, a_n) < K$, where $dist$ means the geometric distance of the two RBSs and K is a constant depending on the average distance between the sites and the diameter of the network. This modification filters out the extreme changes in the topology, which helps to find the near-optimal RBS distribution in much shorter time.

1.7.4 The local improvement process

Although we optimize the RBSs on $level_1$ rest of the network is not optimal, because of the greedy connections of the rest RBSs. In the following we present two additional subprocesses to improve the quality of the final solution. In the first phase of the local improvement, the algorithm iteratively repeats the following two

possibilities, corresponding to the topology constraints, and selects the better one in each step:

- Trying to connect an RBS and the subnetwork is connected to it to another RBS. If there is any possibility to decrease the cost by linking subnetworks together, this process is able to find the suitable changes.
- Trying to swap two links. It means that if there are two links between RBSs x, y and w, v , after the swapping, they will connect RBSs x, v and w, y . This process examines the effect of swapping for almost all possible link pairs (except the changes of the extremely far links). If the network contains unnecessary cross links this process will find and modify them.

The two above subprocesses are repeated as long as they can decrease the cost. Because of the typical small size of the subtrees connected to RBS on $level_1$ it is possible to compute the global optimum inside all such subtrees using exhausting search in reasonable running time. In the second phase of the local improvement process the optimization of these subnetworks is done. The upper bound for the steps required to compute the global optimum of a subtree is the product of the two following equations:

$$C = \prod_{l=0}^{L-1} \frac{(\sum_{k=l+1}^L D_{RBS}^{k-1})!}{D_{RBS}^l! \cdot (\sum_{k=l+2}^L D_{RBS}^{k-1})!} \quad (1.8)$$

$$D = \prod_{m=1}^L \left(\prod_{n=1}^{D_{RBS}^m} \binom{D_{RBS} \cdot n}{D_{RBS}} \right) \quad (1.9)$$

Let $Cost_{subtree.i.best}$ be the lowest cost was found so far for a subtree connected to RBS_i . We interrupt the exhausting search in that phase of the current step, when $Cost_{subtree.i.curr} > Cost_{subtree.i.best}$. Using this method the average required step number to compute the global optimum becomes two order of magnitude smaller than $C \cdot D$. In spite of this I have to note that when $L \geq 4$ and $D_{RBS} \geq 3$ the using of this exhausting search procedure is very time-consuming, therefore in that case it can be skipped from the optimization process.

After the local improvement, all subnetworks connected to RBSs on $level_1$ will be optimal, so the quality of the result will depend only on the quality of the simulated annealing based RBS distribution. On the basis of many tests, we may say that the simulated annealing works very well, consequently the cost of the resulting network will be close to the global optimum.

1.8 Method for Planning UMTS Networks (MPUN)

As we have seen the two well-definable tasks of planning UMTS access networks are the determination of RNC positions and the construction of RBS trees. Beside my proposed methods several known algorithms are usable for solving both tasks. It is not simple to examine and decide what kind of combination of these methods results the best final network. Therefore I present a planning methodology and frame for UMTS networks that is able to integrate my proposed methods and other existing algorithms. In this platform most of the existing algorithms can be compared to each others. The aim of this Section is to find the best combination of methods for solving Task 1 and Task 2 to obtain near-optimal final solution. I have examine the performance of some known algorithms and their combination and I have found that the following possible combinations of them gave the most valuable solution:

1. Method #1: Using the Integrated Planning Algorithm (IPA) proposed in [C1]. In case of IPA the tests show that the bottleneck is the Greedy tree construction method, because it worked not very effective way, caused by the length and capacity dependent link functions.
2. Method #2: Using any kind of known distance-based clustering technique [2] for RNC localization problem, then the mentioned Greedy algorithm is used for tree construction. In this method existing clustering algorithms are tried out, but the problem was with them that they cannot handle the complicated node and link cost structure in an effective way. In most of the cases the resulted RNC positions and clusters do not fit to the further tree construction.
3. Method #3: Using any kind of known distance-based clustering technique, then TCA is used. Here TCA is already coming into the picture, and the tests showed that TCA improves the final solution in a measurable way.
4. Method #4: Using RLA with cost-calculation version 1, then TCA is used. Here the RLA is also already used, with the simplest cost-calculation method (the RBSs are connected to the closest RNC). It is interesting, but the tests proved that this simple cost-calculation results near-optimal RNC locations and clusters for the further optimization of TCA.
5. Method #5: Using RLA with cost-calculation version 2, then TCA is used. Here in each step of RLA the Greedy method is used to determine the clusters, then these clusters are re-designed by using TCA.
6. Method #6: Using RLA with cost-calculation version 3. Of course this configuration is able to obtain the best final solution, since in this case in

each step of the RLA, TCA is used to build up the access trees for all RNCs. On the other hand this is the most time consuming method.

1.9 Performance Analysis

In this section the goodness of the proposed algorithms is examined. Although the real optimum can be computed using some linear programming softwares, it works only on very small input networks (up to 10 RBSs). Because of the great state space, the exhausting search methods are not usable to compute the global optimum in case of practical network sizes, either. A possible way to examine the performance of the methods is the comparison to other, previously proposed ones or calculating lower bounds.

1.9.1 Comparison of different planning methods of UMTS networks

In this section I will compare the different kind of network scenarios that were mentioned in the frame of MPUN. I examined the difference between the results of the algorithms in case of different network sizes and topologies. First the cost of the final solutions were calculated, then the relative cost values were compared (the best solution was 100%, all the other results were compared to this value). Because the algorithms use randomized operations the results are the averages of 100 runs in case of each network.

Table 1.1 shows the relative cost of the networks that can be obtained using the different planning scenarios. In Table 1.2 the relative running times are presented.

Table 1.1: Comparison of different planning methods

#Nodes	Method #1	Method #2	Method #3	Method #4	Method #5	Method #6
100	108.82%	109.76%	107.43%	100.36%	102.54%	100%
200	109.23%	109.92%	106.87%	100.45%	103.34%	100%
300	110.23%	110.54%	107.67%	100.98%	103.45%	100%
500	110.76%	110.57%	107.74%	100.78%	104.12%	100%

It seems that best solution can be obtained by method "6", but I already mentioned that this is very time-consuming method, because TCA must be ran in all clusters in each step of RLA. We can obtain good quality solution during reasonable running time, if we use method "4", since the difference between the results of method "4" and "6" is less than one percent even in cases of great networks. It is important to note that method "4" obtains better results than

Table 1.2: Comparison the running time of the planning methods

#Nodes	Method #1	Method #2	Method #3	Method #4	Method #5	Method #6
100	100%	125.27%	167.87%	235.83%	287.46%	5792.4%
200	100%	123.88%	172.36%	237.53%	279.54%	5896.2%
300	100%	121.65%	177.74%	237.96%	285.06%	5767.6%
500	100%	117.67%	178.97%	243.52%	285.84%	5893.5%

method "5". (Method "4" does not consider the actual tree constraints, while method "5" does it, therefore it would be expected that method "5" should obtain the better results.) The reason of this interesting thing can be that the clustering process of method "4" (direct connection of each RBS to the closest RNC) may result a topology that fits better to the further optimization by TCA. As we see the IPA and the combination of known clustering and greedy tree construction methods obtain measurable worse results than my proposed methods (the difference is about 8–10%). This proves again that it is reasonable to use my proposed planning methods (method "4", "5", and "6"), because they result significant better solutions than the existing methods. If we consider that the most important measure factors of a planning method are the quality of the final result and the reasonable running time, then method "4" is offered for practical network planning problems. If the running time is absolutely not a critical factor then method "6" is proposed. Finally I would like to mention that in case of the best MPUN configuration contains both my proposed methods, which prove that better final results can be obtained using them than using wide range of known methods.

1.9.2 Examination of the RNC Localization Algorithm (RLA)

In this section I will examine the performance of RLA comparing it with a distance based clustering method [2] and the clustering process of IPA and a mixed linear-programming based solution. I examine the special case, when all the RBSs are connected to their home RNC directly in a star topology and the cost function is a simple linear one. In this case the building of RBS-tree is skipped from the problem, so we are able to test only the clustering methods in themselves, how effective they can find the near-optimal number and placement of RNCs. The mixed linear programming-based problem is formulated in the following way:

$$\min\left(\sum_{ij \in E} Cost_{link}(ij) \cdot z_{ij} + \sum_{i \in S} Cost_{RNC_i} \cdot v_i\right) \quad (1.10)$$

subject to

$$\begin{aligned}
0 &\leq z_{ij} \leq 1 && \forall i, j \\
0 &\leq v_i \leq 1 && \forall i \\
\sum_j z_{ij} + v_i &\geq 1 && \forall i \\
\sum_i z_{ij} &\leq N^2 v_j && \forall j
\end{aligned} \tag{1.11}$$

Table 1.3 summarizes the results using different size networks and traffic patterns for the extensive examination of my method. The obtained results are the average of 100 runs in case of each network and algorithm.

Table 1.3: Comparison of different planning methods

#Nodes	RLA	IPA	Distance based clustering	Exact optimum
30	751.142	751.9	755.46	750.645
50	1108.63	1110.35	1123.57	1103.21
100	3790.12	3833.62	3961.38	3772.73
200	9147.2	9251.7	9674.9	9105.8
250	11284.1	11455.5	11982.9	11219.3
300	14287.4	14279.5	15368.4	-
500	23574.3	23864.8	24967.3	-

As we can see the result show that the RLA method can obtain the best solution in most of the cases. In case of smaller size networks the exact optimum can be computed using linear programming, and the results show that the RLA is able to find almost optimal solution. Further tests prove that the results that can be obtained by RLA is the best-fitted results for the following optimization with TCA, therefore it is offered to use RLA for clustering, because it can provide good clusters for the further cost-optimal tree construction.

1.9.3 Examination of the RBS Tree Construction Algorithm (TCA)

I also analyzed the performance of the planning method by comparing the result of TCA and the greedy tree construction method [C1] using different networks. Results of the comparison are shown in Table 1.4.

The tests show that TCA can obtain results about 8.25% lower cost than the greedy tree construction. It is also important to note that TCA works well even in case of great network sizes. If we see, the variance values of the TCA's results are also always small, what proves that the method works very stable. The advantageous variance values due to the post process method, which is able to

Table 1.4: Performance of TCA compared with the Greedy method

#Nodes	Greedy	TCA	Improv.	Variance of TCA
50	10587.3	9340.77	11.8%	0.147%
70	13930.6	12365.4	11.2%	0.383%
100	18252.8	16742.5	8.3%	0.405%
120	21901.9	19883.4	9.2%	0.672%
150	26362	24300.8	7.8%	0.715%
200	33657.4	31434.8	6.6%	1.086%
300	48341.1	45245.4	6.4%	0.972%
500	77144.9	73627	4.6%	1.215%

construct final solution with near similar cost from different output of the previous methods. Therefore, we may say, it is reasonable to use TCA in case of practical network planning problems, where the task is to build up a multi-constrained tree.

Another possible way to examine the efficiency of the proposed algorithm is to relax the topological constraints on the tree and build up a minimum cost spanning tree without any degree and cascading restrictions. This task can be solved using any kind of well-known minimum cost spanning tree algorithm, and this way we can obviously obtain a lower bound on the cost. Because TCA is a heuristic method the variance of its results was computed, and denoted by *Variance of TCA*. The results are given in Table 1.5.

Table 1.5: Lower bound calculation

#Nodes	TCA	Min Span Tree	Error
50	9340.77	9050.1	3.2%
70	12365.4	11949.8	3.4%
100	16742.5	15775.8	5.8%
120	19883.4	18583.8	6.5%
150	24300.8	22565.1	7.1%
200	31434.8	29192.5	7.1%
300	45245.4	41440.5	8.4%
500	73627	65554	11%

The results show that the error of the proposed method was within the range of 3.2-11% on these network examples. In case of smaller networks the lower bound calculation is enough to draw conclusions regarding the quality of the algorithm, but in case of networks with greater sizes the bound is not very tight, so we do not get too much valuable information about the performance of the algorithm.

On the basis of the above test we can say that the proposed algorithm gives very reasonable results in case of practical size networks. Furthermore, if the network size is smaller than 120-150 sites the results will be very close to the global optimum.

1.10 Conclusions

In this chapter I have proposed a novel network model for the design of UMTS Terrestrial Radio Access Networks (UTRAN) and heuristic methods that capable of planning this part of the UMTS network in a cost-optimal way. I have separated the problem into two phases, namely RNC localization and tree construction. For the first problem I proposed a genetic algorithm based method (RLA), the second task is solved by an improved simulated annealing method (TCA). Although the presented algorithms are working in UMTS technical background, equipment constraints and specific cost functions, the *optimization model and process* is able to work in case of any other type clustering and multi-constrained capacitated tree optimization problem with non-linear cost functions. I have tested RLA and TCA extensively, and on the basis of these tests and lower bound computation, I have found that the design methods give results close to the global optimum, even in case of several times 100 RBSs and the variance of the result by running the algorithm more times is very small. I have also examined some possible ways of planning UMTS access networks using different combination of some existing methods and my proposed algorithms. I have shown that my proposed methods obtain the best solution in cases of any planning scenarios. The easy configuration opportunity of the methods provide that an equilibrium between the quality of final solution and the running time can be adjusted in a simple way.

Chapter 2

Cost-based Planning of UMTS Core (Backbone) Networks

2.1 Introduction

Building new UMTS (Universal Mobile Telecommunication Systems) based network seems to be on time. Consequently, some open planning problems appear that related to the cost-optimal design of UMTS core (backbone) networks. Because the core networks are rather mesh-like and the switching points are typically pre-defined, the design tasks contains traffic engineering and dimensioning problems in addition to classical topology optimization. A strong technological constraint that what kind of routing protocol is used in the core network. Furthermore, in the core network the traffic patterns are much more complicated than in the access layer. Another important issue that the volume of traffic is significant greater, which means that a node or link fault can occur critical traffic loss. Therefore we have to take some protection/restoration related questions into account during the core network planning. Because the network equipment in the core network can be more complicated than in access network, the modeling and cost description of the elements are also not very simple task. The cost function of the links is step-like, but in several cases some other factors must be taken into account. It is possible that some part of the core network have already been built, and the task is the cost optimal extension of this existing network. A strong UMTS specific slice of the problem is the design of the optimal connections between the core and access part of the network.

Basically, two main type of design task can be distinguished which appears in UMTS core networks.

- Node and Link Localization Problem (NLLP): In this case the task is to find the positions of the transport nodes in the core network, furthermore it is required to find the locations of the so-called edge nodes, that connects the

RNCs to the transport nodes. The problem contains the planning of the interconnection between the transport nodes, and in some cases the interconnection of the RNCs can be also part of the problem. The other slice of problem is to route the traffic flows, which is not trivial task, because of the step-like link cost functions.

- **Link Localization Problem (LLP):** In this case the positions of the transport and edge nodes are given, the task only the planning of the network topology and the traffic routing.

Because any of the two above task contains subproblems, that are \mathcal{NP} hard in itself, the use of problem specific or general heuristics is required. In [14], [15], [16] and [C10] the authors propose some algorithms for some subproblems that are part of the mentioned planning task, but there is no known algorithmic solution for the whole UMTS Core Network design process. My aim was to fill this gap proposing a planning method that is able to handle the specific feature of this problem.

2.2 Architecture of the UMTS Core Network

In Figure 2.1. we can briefly overview the structure of UMTS core network and the role of the different network elements. This core network model is conform to the current UMTS Releases, but the model is general and it can be extended for support further network architectures.

The Core network contains three main types of equipment, namely the Transport Nodes (TN), the Media Gateways (MGw), and the RNCs. The Transport Nodes must manage and route the traffic in the core network, as well as manage the MGws. The role of a Media Gateway is to provide the connection between core network and the UTRAN, and handles lower layer function for both packet and circuit mode communication. It handles user data and mapping media flows as well as switching and routing between core and UTRAN; to perform these functions the MGw contains an IP router, an ATM switch and a media stream handler. As we can see, structurally, the RNCs are part of the UTRAN, because their task is to manage the radio channels of RBSs connected to it. Otherwise, they must also concentrate the traffic flows of connections and trunk them to the upper level core network and handle the soft handover traffic. The known UTRAN design algorithms give only approximative solution for the interconnection of the RNCs and does not solve the problem of MGw placement. Since MGw belongs to core network, therefore I handled the RNC interconnection task together with core network design. The Transport Nodes (TN) in core network can be different kind of switching or routing elements according to the type of the core network. If the core network is an MPLS domain, then the TNs are Label Switching Routers (LSR), while if the core network uses ATM technology, then the TNs contains

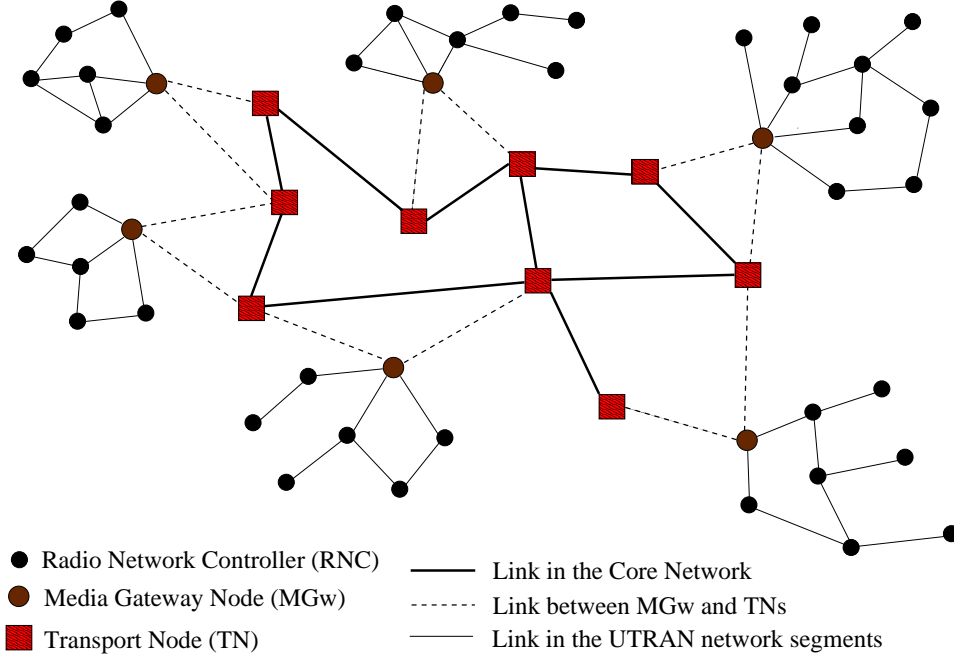


Figure 2.1: UMTS network architecture

ATM switches. If the core is a pure IP network, then the TNs are routers and the routing of demands happens according to the OSPF (Open Shortest Path First) routing rule. Because of the possible different transmission protocols the proposed algorithm was designed to handle all of them.

2.3 Problem Statement

In this section the used network model and the cost functions are defined.

Let N denote the set of nodes, E denote the set of possible links and D denote the set of demands in the network. Let $\sigma_e = 1$ if link e is installed into the network, otherwise $\sigma_e = 0$. Let cap_d is the value of the d th demand. As we can see on Figure 2.1 the network to be planned can be divided into two main layers from the viewpoint of modeling. One layer consists of the MGWs and the TNs, while the another one consists of the RNCs of the different UTRAN networks. Because of this special architecture of the network we define some additional sets for different kinds of subnetworks and network nodes. Let $S = \{S_1, S_2, \dots, S_U\}$ denote the set of UTRAN networks, and let $R^{S_i} = \{r_1^{S_i}, r_2^{S_i}, \dots, r_M^{S_i}\}$ denote the geographical points of the RNCs in UTRAN area S_i , $1 \leq i \leq U$. The number of UTRAN networks denoted by $|S|$, while the number of RNCs in S_i is denoted by

$|R^{S_i}|$. Let $B = \{b_1, b_2, \dots, b_B\}$ denote the set of TNs in the core network.

Let y_n is 1 if a MGw can be installed into node n and 0 if MGw installation is prohibited. Similarly, let x_n is 1 if a MGw is installed into node n , otherwise 0. From the viewpoint of the cost we can distinguish node and link cost. For the nodes a step-like cost function (denoted by $Cost_{node}(n)$) is used, which consists of two types of subcost, namely:

- Equipment cost, which represents the installation/investment cost of the equipment and required subdevices (namely subracks, processors, boards, etc) according to the traffic passed the node.
- Port cost that depends on the number of incoming and outgoing ports and the required capacity of the current port.

For the links, the cost has a capacity dependent and a capacity independent part. The capacity dependent part is typically represented by a step-like function according to the available link capacities and denoted by $Cost_{link}(C_e)$, where C_e is the sum of traffic that passed link e . I introduce the following notations to help handling the step-like function. Let c_e^l denote the capacity of link e where the l th step is installed in the network; $l \in \{1 \dots L\}$, where L denotes the last step. For simplicity, l will be called *the step index* on the link e . The capacity independent part of the link cost is proportional to the distance of the nodes and handles the cost of the repeaters along the link (if any). This cost can be described by the value of the first step of step-like function ($Cost_{link}(c_e^1)$).

Of course, the model enables to apply different cost functions for different links and nodes, but practically three main cost functions are used for the node (RNC, MGw and TN) and two main cost functions for the links (link between RNCs, or RNCs and MGw, as well as link between TNs, or TNs and MGw). Varying the cost parameters, the algorithm can consider the existing network parts (reduced node and link cost; typically the installation cost is zero for the existing equipment).

This way the total network cost can be calculated:

$$Cost_{total} = \sum_{n \in N} Cost_{node}(n) + \sum_{e \in E} Cost_{link}(C_e) \quad (2.1)$$

The ultimate goal of the optimization is to minimize the $Cost_{total}$ value, subject to the following constraints:

- $\sum_{n \in R^{S_i}} x_n = 1$ for all S_i
- The network is fault-tolerant (at least two-connected) between the MGws.
- Optionally, the network is also fault tolerant between the RNCs and their home MGw.

- All traffic demands are satisfied, and the network is consistent (there is no overloaded link in the network).

2.4 The Solution

Because the afore-mentioned problem is \mathcal{NP} -hard (it can be traced back to the well known Steiner tree problem), therefore it is necessary to use of heuristic method. Otherwise the state space of the problem is great, therefore I decomposed the original task into smaller, but more well-defined subproblems and I solved the original problem in two different phases. Although the problem decomposition leads to the degradation of the final result, but the tests showed that the two-phase optimization can result good solution and the running time is very favourable.

Let us overview the subproblems to be solved in the phases.

- Phase 1: The task is to find a near-optimal placement of MGws and find reasonable network topology using linear cost function instead of the step-like one. In this way the complexity of the problem can considerable be reduced. Because the linear function has an $A_e + B_e * C_e$ form for each link e , it is required to calculate A_e , which related to the installation cost of a link, and B_e is the gradient of the function, which is related to the capacity and cost steps of the original link cost function, while C_e is the sum of traffic passed on the link currently. A Simulated Annealing based heuristic algorithm is used here which is able to solve the placement of the MGws and determination of the link positions between both the RNCs and the TNs. In this phase the goal is to minimize the $A_e + B_e * C_e$ function. (THESIS 2.1)
- Phase 2: In Phase 2, the original (step-like) link cost function is used. The network topology has already been not modified, but a heuristic method is used to re-route some demands caused by the step-like cost function, to reduce the original network cost. For solving Phase 2, I proposed two kind of algorithms, the first one based on Simulated Annealing, while the second one is based on an improved version of Michal Piòro's Simulated Allocation algorithm. Here the goal is to minimize the original cost function $Cost_{total}$. (THESIS 2.2)

I have also proposed a so-called PostProcess method, which is a collection of local improvement methods that try to modify the paths of some demands in order to obtain a better network configuration. PostProcess can be used effectively after finishing Phase 2. During the PostProcess phase $Cost_{total}$ is to be minimized. (THESIS 2.3)

In the following two subsections the proposed methods applied in the two phases are outlined, while Section 2.5 overview the most important features of the Post-Process method.

2.4.1 Solving Phase 1

In the initial step of the optimization we have to calculate the values of A_e and B_e for each link e . Because A_e models the installation and distance dependent cost of a link it is equal to the first step of the step-like cost function, so

$$A_e = Cost_{link}(c_e^1).$$

Computing B_e value is a little bit complex, because there are some ways how we could characterize a step-like cost function by a linear one. I selected the following method to calculate the B_e value:

$$B_e = \frac{Cost_{link}(c_e^L) - Cost_{link}(c_e^1)}{c_e^{L-1} + \beta(e) \cdot (c_e^L - c_e^{L-1})}$$

, where $0 < \beta(e) \leq 1$ and $\beta(e)$ can be used to slightly modify the gradient of the current link function. By the help of $\beta(e)$ it can be adjusted where the linear function intersects the final step of the step-like cost function. On the basis of test the best values of β are between 0.4 and 0.6.

After the calculation of linear cost function parameters for each link, the optimization process can be started. Here I use an improved version of the classical simulated annealing (SA) meta-heuristic, which is quite effective for solving this kind of optimization problems [10] and [11]. SA starts out with a solution denoted by x^{old} . (The initial solution is generated randomly.) Then, at every step, the algorithm selects a neighbour of x^{old} , denoted by x^{new} , using function $Neighbour(x^{old})$. Then a stochastic acceptance criteria is applied to decide whether state x^{new} is acceptable or not. If yes, then the optimization is continued from x^{new} , otherwise using $Neighbour(x^{old})$ another x^{new} is generated. The effectiveness of the SA is determined by how we define the state space of the optimization and the function $x^{new} = Neighbour(x^{old})$. In this case a state is a set of links that is established into the network, and the position of each MGw. This state space provides, that the algorithm is able to find optimal solution, since it is the optimal position of the links and MGws.

The most important steps of the algorithm can be seen below:

Initialization step. Calculation of A_e and B_e values for each link e . Some links are installed into the network and the demands are routed on the distance-based shortest paths.

Step 1. A new network topology (a neighbour state of x^{old}) is obtained in one of the following ways:

- Link switch off or on in the UTRAN level (LSU): In this step a UTRAN network r_i and a link e – which is contained by r_i – is selected randomly, and its status will be: $\sigma_e = 1 - \sigma_e$.

- Link switch off or on in the core level (LSC): A link e is selected randomly from the CN (the end point of the selected link must be MGw or TN) and its status will be: $\sigma_e = 1 - \sigma_e$.
- MGw movement (MGM): An UTRAN network r_i is selected and the MGw will be moved to a new node inside the UTRAN area. Because the traffic demands must leave the UTRAN using MGw, all traffic flows that originate and terminate r_i must be rerouted using the new place of MGw.

Step 2. After each modification all demands must be routed using Dijkstra's method with the following weight function w_e for the links:

$$w_e = \begin{cases} B_e * C_e & \text{if } \sigma_e = 1 \\ M & \text{if } \sigma_e = 0 \end{cases}$$

M is a great number ($M \approx (2..5) \cdot A_e$). This weight function enables to use those links that are not installed into the network, yet (otherwise we have to handle the problem of network connectivity in each step, which is a very time-consuming process). If any demand wants to use a link which is not installed, then its extremely high weight (M) will force the algorithm to use more installed links into the network instead of the prohibited one (if possible). After routing the network cost is calculated, what is equal to the sum of link initialization costs ($\sum_{e \in E} A_e$) and traffic dependent cost of the links ($\sum_{e \in E} B_e * C_e$). If the new network state is accepted then σ_e will be 1 for all the links that carries any traffic, and the optimization will continue from this state at *Step 1*. until pre-described number of iterations.

I have some notes on adaptivity of our simulated annealing method. In each step it is examined, whether the current modification was successful (it resulted the decreasing of network cost) or not. If yes, then the probability of the current step (one of from LSU, LSC, MGM) is increased, while the probability of two different type movements are decreased. If the current step was bad, then the probability of the movement is decreased.

2.4.2 Solving Phase 2

The algorithms that work in Phase 2. start out from the output of the previous method and try to optimize the paths of traffic demands using the step-like cost function. In this phase the network topology has not modified any more (except when a link is not used; this link will be deleted from the network) only the optimal routing of demands are searched using the original step-like cost function. Two algorithms are proposed for solving this task, the first one is based on Simulated Annealing and the second one is based on Simulated Allocation.

The Simulated Annealing (SAN) based algorithm

In this method a pre-defined set of possible different explicit working and protection paths are searched for each demand. Firstly, all (or some of) the shortest paths are searched (their length measured in hops and denoted by L_{sp}), then the $L_{sp} + 1$ length paths are searched, and so on. The results have shown that $L_{sp} + 3$ length paths are enough to obtain a near-optimal solution. Although the pre-definition of the paths decreases the state-space, but tests show that such of results that contain longer paths than $L_{sp} + 3$ are extremely rarely, because those may increase the load of the current demand in the network, resulting cost increasing, too.

The k th path of demand d is denoted by p_k^d , $k = 1 \dots K$, where K is the number of different paths. The state of the optimization is which path is selected to carry the demand. The neighbour states can be obtained by selecting a demand d randomly and moving it from p_k^d to p_l^d , where $l \neq k$. Then the cost of the new state is computed and on the basis of stochastic acceptance criteria of simulated annealing, it is decided whether the new state is acceptable or not. The algorithm finished its operation after pre-defined iterations.

Because the paths of demands are determined in advance the sequence of demand allocation does not influence the goodness of the final solution. Practically it means that in one step of the optimization all demands are allocated simultaneously. Further important feature of this method is that the working and protection paths are handled independently during the optimization. My tests show that this handling of working and protection paths are more effective than handling them together as path-pairs.

The Simulated Allocation (SAAL) based algorithm

This method is an improved version of the algorithm proposed by Michal Pióro in [13]. The algorithm is based on iterative demand allocation and deallocation. A probability value ($1 > pr > 0.5$) is defined and in each step a random number $a \in [0, 1]$ is generated. In the original version if $a < pr$ then a demand is routed in the network, otherwise an allocated demand is selected randomly and deleted from the network. The process is stopped when all the demands are allocated. Usually the routing of demands happens using simple Dijkstra's method, rarely extended with a kind of capacity related link weight function. The original version of the Simulated Allocation obtains acceptable results, but I improve the method in some points to obtain better results.

Instead of the capacity dependent weight function I used a function that depends on the marginal cost of a demand allocation. I found that this kind of function fits better to the cost-based optimization. The weight function I used for allocation of demand d is the following:

$$w^d(e) = \begin{cases} \epsilon + \left(1 - \frac{(c_e^l - C_e + cap^d)}{c_e^l}\right) & \text{if } C_e + cap^d \leq c_e^l \\ Cost_{link}(C_e + cap^d) - Cost_{link}(C_e) & \text{if } C_e + cap^d > c_e^l \end{cases}$$

, where ϵ is a small number, cap^d is the value of demand d , and c_e^l is the current capacity of link e (the l th step according to the step-like link capacity function).

In the classical SAAL only one demand is allocated or de-allocated in one iteration step. In my version there are two states. In the allocation one several demands are selected and allocated one by one. The demands to be routed can be selected randomly or sorted in an increasing order or sorted by the marginal cost of their allocation at the current network state. In the de-allocation state one or more random selected demand(s) is deleted from the network in the following ways:

- One demand is deleted that contains a random selected link: This is the basic step of the classical SAAL method. In my improved version this deallocation mode is not very significant, so the probability of using this way to delete a demand is small.
- One demand is deleted from several most-loaded links: The role of this step is to obtain a network state, in which there are few very loaded links, and the overall network load is uniform.
- About 50 percent of the demands are deleted from the most-loaded link: This is a rather drastic step to move out the network from the current state, because in this step some demands are deleted, consequently the network load is decreased significantly. This step helps to find a better path for several demands that used not optimal path earlier.
- Those demands are deleted that result the largest cost reduction: The non optimal configuration of several demands results higher network cost. This step helps to find these demands and delete them from the network.

The repeating setup and tear down of the demands provides that great part of the state space will be examined. The probability to select one from the above alternates can be fixed, or can be adjusted by the algorithm time to time according to their efficiency.

In case of protection, the primary and backup paths handled together (if a demand is selected to be routed both primary and backup paths must be allocated for it).

The classical SAAL finishes its operation when all the demands are allocated. I proposed an iterative more-phase process, which consists of several classical SAAL processes. After each phase, when all demands are allocated some of them

are deleted from the critical part (bottleneck links, or under-/overloaded parts) of the network. The percent of deleted demands and the probability of allocation decreases phase by phase as it is illustrated on Figure 2.2. This iterative process is more time-consuming comparing to classical SAAL, but provides about 4-6% better results.

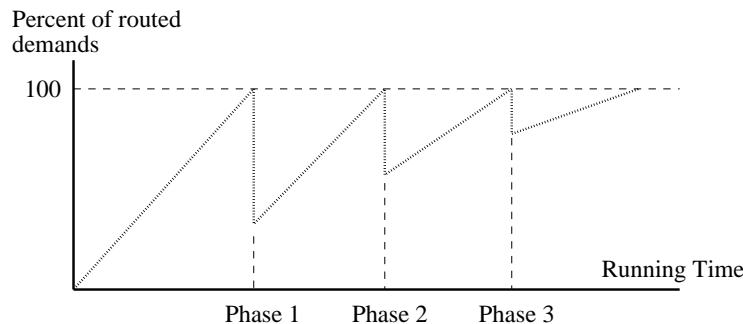


Figure 2.2: UMTS network

In case of classical SAAL the probability of demand allocation is fixed during the optimization. I proposed a possible way for adaptive probability adjustment:

- when the demand allocations are successful the probability is increasing for faster configuration.
- when some demand allocations are failed, it can occur a local optimum, therefore the probability is decreasing in order to delete more demands from the network to leave the current network state.

Using this adaptive probability adjustment further 2-3% better results can be obtained.

2.5 Post-Processing: Further Improvements on the Solution

The outlined basic algorithm usually obtains favorable results, but often by using simple local subprocesses helps to achieve a better solution. In this section four methods that can improve the result of the basic algorithm are presented. All methods are based on iterative deallocation and reallocation of some demands. The following notations are using here:

- Let l be the integer, for that $c_e^{l-1} < C_e$ but $c_e^l \geq C_e$.
- Let $\delta_e = c_e^l - C_e$. It is the current residual capacity on link e .

2.5.1 Capacity decreasing method

The idea behind this method is that by re-routing some demands the capacities of low utilized links can be reduced. Thus it is possible to achieve that on some links there will be less capacity installed while on some links might be more, but the total cost of the network reduces.

- Step 1 'Link selection': In the first phase of this process C_e and δ_e are computed for all links. Then that link f is selected, where δ_f is minimal. The demands that should be re-routed from the link f in order to install a cheaper link (a link having step index one less than it had originally), are listed and arranged into increasing order according to their demands. These demands are removed from the network.
- Step 2 'Traffic re-routing': The values C_e are calculated again. From the set of demands to be re-routed the greatest one is selected and re-routed using the weight function in 2.5.2, which tries to force the current demand to avoid the selected link f . Then this demand is deleted from the set of selected demands.
- Step 3 If the set of demands to be re-routed is not empty then the process continuous at Step 2, else it moves to Step 4.
- Step 4 If the re-routing of all selected demands results in a smaller network cost, then the process starts again at Step 1 otherwise it is stopped.

2.5.2 Residual capacity based re-routing

Sometimes the routing of some demands is not optimal. This process helps to check it and tries to re-route all demands one by one, so that it results smaller cost. A demand d is removed from the network and the values C_e are calculated again. Then the demand d is re-routed using the weight function $w^d(e)$ for link e :

$$w^d(e) = \begin{cases} \frac{(C_e + cap^d) - c_e^{l-1}}{c_e^l - c_e^{l-1}} & \text{if } C_e + cap^d \leq c_e^l \\ Cost_{link}(C_e + cap^d) - Cost_{link}(C_e) & \text{if } C_e + cap^d > c_e^l \end{cases}$$

Using this kind of weights, demand d follows its original path or it is re-routed resulting lower network cost. There are several possible options for sorting the demands according to its value (cap^d), the length of its original path or the linear combination of these two factors.

2.5.3 Link capacity extension

The idea behind this process is that increasing the capacity of a link can cause capacity reduction on other link(s), resulting smaller network cost. If the current step index l on a link f is not the greatest possible one, then the link capacity can be increased and the link can be replaced by a link which has capacity c_f^{l+1} . Then all the demands are re-routed one by one. For demand d , first it is removed from the network and the values C_e , $e \in \{E \setminus f\}$ are re-calculated, then it is re-routed using the weight function mentioned in 2.5.2, but for link f the following function is used:

$$w^d(f) = \begin{cases} 0 & \text{if } C_f + cap^d \leq c_f^{l+1} \\ Cost_{link}(C_f + cap^d) & \text{if } C_f + cap^d > c_f^{l+1} \end{cases}$$

It means that this capacity increasing is free of cost during the re-routing of the demands. After each demand is re-routed, the cost of the network is calculated. If the total cost of the modified network is less than the original one, then the expanded link remains in the network, else the original capacity (c_f^l) is set back on link f and the demands will follow their original routes. The process is finished when all links have been examined. The sorting of the links can be done randomly (it is possible that a link is selected more than once) or according to its extension cost ($Cost_{link}(c_e^{l+1}) - Cost_{link}(c_e^l)$), furthermore according to the value of $C_e - c_e^{l-1}$ or $c_e^{l+1} - C_e$.

2.5.4 Link deleting

Sometimes the network cost can be decreased significantly by deleting an existing link e from the network. Firstly, the links are sorted according to their initialization cost ($Cost_{link}(c_e^1)$), then the most expensive link is deleted from the network and the demands that used this link are removed from the network. Then they are tried to be routed in the modified network one by one, but before each attempt the values C_e are calculated. For the re-routing process I use the cost function given in 2.5.2. If there is a demand that cannot be satisfied then the link is restored. If all demands can be re-routed and the cost of the new network is smaller than the original one, then the link is removed permanently. The process is stopped when no more link can be deleted.

2.6 Application of the methods to solve general network planning problems

Although the above methods are developed for the specific task of UMTS core network design, they can be applied for classical topology optimization tasks as

well.

In case of *link localization problem* the nodes are given and fixed, and the problem is to find a cost-optimal link set for connecting the nodes. In this case only the LSC movement of the Phase 1 algorithm is used for topology optimization, the two others are prohibited. If the problem is the *transit node and link localization*, where the network contains some transit nodes that do not generate traffic, their role is to connect the access nodes as a backbone network. (In the UMTS interpretation Transport Nodes could be the transit nodes, while RNCs correspond as access nodes). Thus, the previous task is completed with the selection of some nodes from the transit node set that form a backbone network, which connects the access nodes to each others. In case of this task LSC and MGM movements are used, but MGM can operate on any subset of nodes. For the traffic routing task both the Phase 2. algorithms are usable without any restriction. These methods can be used for traffic routing problems in any kind of networks without any modifications. The PostProcess is also can be applied to obtain a better network topology and/or more advantageous traffic routing without any limitations.

2.7 Performance Analysis

In this section a brief performance analysis of the proposed algorithm is outlined. For the tests 5 random generated networks were used; the properties of the networks are summarized in Table 2.1.

Table 2.1: The test networks

Networks	Number of nodes in UTRANs	Aver. number of UTRANs	Number of TNs
A	5	8	10
B	5	15	10
C	10	8	15
D	10	15	20
E	15	15	25

Because of the great state space and some specialities of the problem, it is not possible to compute the global optimum in case of practical network sizes (several UTRAN networks with 7-10 RNCs in each one, and 20-25 transport nodes). A possible way is to examine the performance of the proposed method using some kind of lower bound calculation.

2.7.1 Lower bound calculation for the Phase 1 method

We used the so-called Lagrange relaxation method to compute a lower bound for the Phase 1. problem. The essence of the Lagrange relaxation is roughly the following. Since the original problem is too complex to solve directly, we eliminate a set of constraints, hence these constraints can be violated. The advantage of eliminating an appropriate constraint is that the resulting problem can be solved exactly. It can be proved that the solution of the new problem is a lower bound on the original problem. In order to stay "close" to the original problem, for violating the constraint a so-called penalty (depending on the violation) has to be added to the cost. In this case the relaxation was that a demand may use uninstalled links for extra penalty cost. I take the assumption that places of the Media Gateways are given, so the original problem is reduced to link localization problem. The experience was that this simplification does not cause significant quality degradation, the lower bound may be considered to be strict. In case of all test networks, the algorithm ran 100 times. Error means the difference between the average of the results and the lower bound, while variance means the variance of the obtained results. The results of comparison are shown in Table 2.2.

Table 2.2: Evaluation of the Proposed Phase 1 Algorithm

Network	Proposed algorithm	Lower bound	Error	Variance
A	12355.9	11246.8	8.97%	0.57%
B	19256.6	17347.6	9.91%	0.89%
C	28654.3	25324.5	11.6%	1.34%
D	43587.9	38273.8	12.2%	1.57%
E	62358.4	53756.9	13.8%	2.34%

The results show that the performance of the algorithm is quite good, the relative small error values imply that the final cost will be close to the global optimum in almost all cases. The value of variance is also reasonable, therefore we may say that the algorithm works stable even in case of great network sizes.

2.7.2 Comparison of the classical and improved Simulated Allocation based algorithm

The aim of this Section is to prove that my improvements of the classical Simulated Allocation method provide that measurable better solution is obtained. Previous tests in the literature (like [13]) show that classical SAAL works optimally if the probability of demand allocation is about 0.7 ... 0.8, therefore I apply these values. For my improved version of the SAAL works in the best way if the probability is

just a little bit greater than 0.5 (it is usually between 0.55–0.6)). To obtain the variance of the results of the methods, they were ran 100 times for every test networks. The results are summarized in Table 2.3.

Table 2.3: Comparison of classical and improved SAAL

Network	Classical SAAL	Improved SAAL	Variance of classical SAAL	Variance of improved SAAL
A	17892.5	15878.4	5.46%	3.54%
B	26273.4	24789.3	5.78%	4.08%
C	34935.7	32373.6	7.34%	4.74%
D	51782.4	48302.1	8.38%	4.34%
E	71634.6	67137.4	10.73%	5.98%

As we see the improved simulated allocation based algorithm obtains about 6–10% better results than the original version. The second important thing what we can recognize that the variance of the results of improved SAAL is significantly smaller than the variance of the result of classical SAAL. This proves that the improved version provides more stable operation than the original one. Although the running time of the improved version is about three times larger than the original one’s, but in this kind of (green–field) network planning tasks the running time is not very critical factor.

2.7.3 Comparison of the two second phase methods

In this section the SAN and SAAL based methods are compared. It is assumed that the network topology is given and the task is to find a cost optimal routing of the demands according to the step–like cost functions. The results are summarized in Table 2.4.

Table 2.4: Comparison of two second phase processes

Network	SAN method	SAAL method	Runtime SAN [s]	Runtime SAAL [s]
A	15473.6	15878.4	20.15	325.24
B	24567.3	24789.3	30.25	457.38
C	32478.9	32373.6	43.33	643.34
D	49307.9	48302.1	67.87	894.46
E	68327.3	66137.4	103.54	1234.34

It can be seen that the SAN method gives better results in case of smaller size networks. The reason of this could be that SAN can handle the pre-defined paths

independently from each other, while in case of SAAL, the routing of a demand depends on the current status of the network. In case of greater network sizes the SAAL is the "winner", because its adaptive operation is fitted better to the great state spaces. I have to mention that the difference between the result of the two algorithms are not very significant, because it is less than 3%.

2.7.4 Testing the effectiveness of the PostProcess method

In this section I will demonstrate that my proposed post-process methods are able to improve the results of the basic planning algorithm. Because the PostProcess contains only deterministic methods, one run is enough to obtain the reachable best result in case of any network. The PostProcess will be started out from the results of the previous test; the result are summarized in Table 2.5.

Table 2.5: Comparison of two second phase processes

Network	SAN results	SAN result after PostProc.	SAAL result	SAAL result after PostProc
A	15473.6	<i>15183.4</i>	15878.4	<i>14973.5</i>
B	24567.3	<i>24293.3</i>	24789.3	<i>24293.3</i>
C	32478.9	<i>30764.8</i>	32373.6	<i>30278.7</i>
D	49307.9	<i>46879.1</i>	48302.1	<i>46193.4</i>
E	68327.3	<i>63786.7</i>	66137.4	<i>61832.7</i>

Because PostProcess was able to improve the results in case of all networks, it works properly. In case of "Network A" we can see an interesting example for that case, when the PostProcess is able to obtain a better solution from a worse starting one (final solution of SAN was better than SAAL's result, but after the Postprocess they were changed). In case of smaller networks the improvement of the PostProcess is not very significant (around 2%), since the basic algorithms' results are quite good in case of such problem sizes. On the other hand, when the networks are larger the effectiveness of the PostProcess is significantly greater (in case of "Network E" the improvement is more than 9%).

2.8 Conclusions

In this chapter a heuristic planning method is proposed to solve the problem of designing UMTS core (backbone) networks. The following subtasks are solved in the frame of this design problem: task (a) connecting the RNCs of one UTRAN area to each other, task (b) optimal placement of Media Gateway nodes to the border of UTRAN and core network and task (c) connecting the Transport Nodes

(TNs) to each other. Because the state space of original problem is quite great to obtain a reasonable solution the problem was decomposed into two main phases. In the first phase task (a) and (b) is solved using a Simulated Annealing based heuristic. For solving the other phase I proposed two algorithms, one of them also uses Simulated Annealing, while the another is based on an improved version of Simulated Allocation. I have also proposed a PostProcess method, which is able to improve the result of the main algorithms. The tests showed that the proposed planning method is able to handle networks with great size and the obtained results are favourable.

Chapter 3

Optimization of OSPF Administrative Weights

3.1 Introduction

The Open Shortest Path First (OSPF) is the most commonly used Interior Gateway Protocol in today's IP networks. The OSPF uses the shortest paths for routing packets according to the given weights of the links and applies the so-called Equal-Cost MultiPath (ECMP) principle in cases of multiple shortest paths. The OSPF routing procedure is essentially as follows: All nodes maintain a graph representation of the network topology, where the nodes are the vertices and the links are the edges, respectively. Each link e is assigned a positive integer number w_e , weight value, and these values are sent to the network nodes by means of the OSPF link-state flooding mechanism. In consequence, all the nodes are aware of the weights $w = (w_1, w_2, \dots, w_E)$ of all network's links. If a packet arrives at an intermediate node t , and destined for node u , the current node t determines the next hop of the packet towards u using local routing table based on the administrative weights. The shortest paths are calculated on the basis of the current links' weight system.

The route computation at the current node is independent from the packet's originating node and the previous intermediate nodes. If there is more than one outgoing links from node t belonging to the shortest path from t to u , then the traffic is distributed evenly among these links.

I am dealing with *static* weight systems, where the weights do not change dynamically, not allowing for flexibility during normal network operation in a short time range. However it is possible to adjust the weight system and made it fit to the current network state and traffic conditions from time to time, if necessary.

Because the weight system strictly determines all paths, its adjustment is critical for efficient resource utilization. Consequently, to solve the weight system setting problem in the best way might be one of the most fundamental tasks of

the network operator leading to efficient utilization of network resources. We note that for the network of practical size, the proper adjustment of the weights is a very difficult task and therefore we have to use algorithmic approach.

The weight computation related problems can be divided into two basic classes, namely *uncapacitated* and *capacitated* problems. The first class deals with the task of flow allocation problems, because the physical link capacities are not fixed. Heuristic and metaheuristic methods, and even exact methods have been proposed in this area. For more details we refer to [18] and [19] .

In this chapter the *capacitated flow allocation problem* is addressed, (where the capacities of the links are pre-defined) according to the OSPF routing rule. This task is proved to be \mathcal{NP} -hard as shown in [C3], therefore it is necessary to use some kind of heuristic algorithms to obtain feasible solution during reasonable running time.

Basically two main approaches are possible in this problem, namely the one- and the two-phase approach. In case of the first one the weights are calculated directly on the basis of some kind of functions of the link utilization and some other network metrics, during an iterative process. In the other case, first a near-optimal path system is determined according to the related OSPF routing rules, then suitable weights are computed for the existing path system. In [C2] the background and properties of both approaches were outlined.

3.2 Problem Statement

The optimization task to be solved is as follows:

Indices:

$i = 1, 2, \dots, N$	node
$d = 1, 2, \dots, D$	demands
$j = 1, 2, \dots, m(d)$	paths for realising demand d
$e = 1, 2, \dots, E$	links

Constants:

h_d	minimal volume of demand d that must be realized
$v_{edj} = \begin{cases} 1, & \text{if link } e \text{ belongs to path } j \text{ realising} \\ & \text{demand } d \\ 0, & \text{otherwise} \end{cases}$	
y_e	capacity of link e
b_e	value of one idle capacity unit on link e
K	bound for the weights

Variables:

x_{dj}	flow realising demand d on path j implied by the weight system
w_e	weight of link e - integer variable, $w = \{w_1, w_2, \dots, w_E\}$
a_e	allocated capacity on link e

Objective: Maximize residual capacity

$$\text{maximize: } \sum_e b_e \cdot \left(y_e - \sum_d \sum_j v_{edj} \cdot x_{dj}(w) \right) \quad (3.1)$$

Constraints:

$$\sum_j x_{dj}(w) = h_d \quad d = 0, 1, 2, \dots, D \quad (3.2)$$

$$\sum_d \sum_j v_{edj} \cdot x_{dj}(w) \leq y_e \quad e = 0, 1, 2, \dots, E \quad (3.3)$$

$$1 \leq w_e \leq K \quad e = 0, 1, 2, \dots, E \quad (3.4)$$

As it can be seen in (3.1) the objective is to maximize the residual capacity in the network. I should note, however that the structure of the applied optimization procedures are not developed only for the objective function (3.1), any another type network performance metric (for example minimizing the variance of links' residual capacity) can be given as a goal to be achieved. Constraint (3.2) guarantees that all demands are satisfied, while constraint (3.3) provides that links' loads do not exceed links' capacities. The weight system contains limited weight values, according to constraint (3.4). The lower bound guarantees, that weights are positive, therefore there can be no loops in the shortest path. The upper limit is coming from technological constraints.

In the proposed algorithms a general model of the telecommunications networks is used, i. e. a directed graph for both the demands and the physical interconnections. The reason of using directed demands is that in many cases, between two nodes the traffic of opposite directions is very different (such a situation is for example a client - server type connection). We have to use also directed physical network, because it is possible that a demand from a to b follows different path than from b to a . Because the undirected model can be described by the directed

one in a very simple way, the algorithm is also able to handle undirected network if it is required. The demand graph denoted by $F = (N, D)$, where vertices N denote the terminate points of a demand, while edges D represent the traffic demands between two points. The graph of the physical network is denoted by $G = (N, E, Y, w)$, where vertices N and edges E represent the routers and physical interconnections of them, respectively. Y means the capacity vector for the edges, while w denotes the current weight system.

3.3 MIP Formulation

It is important to examine what kind of optimization techniques can be use to solve this problem efficiently. Therefore we found a formal Mixed-Integer Programming formulation of the problem. We consider G , and the given demand to be allocated from node v to node t is given by $d(v, t)$. Below, $o(e)$ and $t(e)$ denote the starting and end nodes of link e , respectively. Let $W(v, t)$ be the length of the shortest path from v to t (variables), and let $(\delta(e, t) : e \in E, t \in N)$ be a set of binary variable such if $\delta(e, t) = 1$ link e is on a shortest path to node t . Let $f(e, t)$ (variables) denote the flow to node t on link e (it should be zero if e is not on a shortest path to node t). Let $f_x(v, t)$ (variables) denote the maximum flow to node t on all links outgoing from node v ; this should be the flow on each link that is on a shortest path to node t .

$$\begin{aligned}
& \sum_{e:o(e)=t} f(e, t) - \sum_{e:t(e)=t} f(e, t) = - \sum_{x \in N} d(x, t) \\
& \text{for } \forall t \in N \\
& \sum_{e:o(e)=v} f(e, t) - \sum_{e:t(e)=v} f(e, t) = d(v, t) \\
& \text{for } \forall t \in N, \forall v \in N, v \neq t \\
& \sum_{t \in N} f(e, t) \leq c(e) \\
& \text{for } \forall e \in E \\
& 0 \leq f_x(o(e), t) - f(e, t) \leq (1 - \delta(e, t)) \cdot \sum_{x \in N} d(v, t) \\
& \text{for } \forall t \in N, \forall e \in E \\
& f(e, t) \leq \delta(e, t) \cdot \sum_{v \in N} d_x(v, t) \\
& \text{for } \forall t \in N, \forall e \in E \\
& 0 \leq W(t(e), t) + w(e) - W(o(e), t) \leq (1 - \delta(e, t)) \cdot |N| \\
& \text{for } \forall t \in N, \forall e \in E \\
& 1 - \delta(e, t) \leq (W(t(e), t) + w(e) - W(o(e), t)) \cdot |N| \\
& \text{for } \forall t \in N, \forall e \in E \\
& \sum_{e:o(e)=v} \delta(e, t) \geq 1 \\
& \text{for } \forall t \in N, \forall v \in N
\end{aligned}$$

If we put equality in the last constraint, we force the shortest path to be unique.

This MIP formulation allows finding global optimum, therefore we tried to solve it using the well-known CPLEX linear program solver. We get valuable results only in cases of very small networks (number of link less than 25–30, and number of nodes less than 12), in cases of larger networks the CPLEX cannot present a solution or the running time was extremely long (average time was about 25 hours). It proves that if we want to optimize practical size networks we need to use heuristic approach.

3.4 Outline of the Simple Weight Adjustment (SWA) Algorithm

In case of the development of this method the most important factor was to obtain a reasonable result during the shortest possible running time. It makes possible to use the algorithm in situations, where on-line weight adjustment problems can be appeared and fast re-configuration is very important (faults, drastic traffic changes and other kind of extra-ordinary cases). The algorithm is based on the *one-phase* weight adjustment approach and consists of two local search procedures:

- Weight adjustment (WA): This method is used when the network is overloaded (there is at least one link which is overloaded) and the weights of the overloaded links will be increased, while the underloaded links' weights will be reduced. The goal is to find a weight system where there is no any overload in the network.
- Load optimization (LO): When the network is underloaded only several selected links' weights are changed to achieve the best network utilization as possible. The goal is to maximize the residual capacity in the network.

The operation of the method can be seen in the following pseudo-code:

Begin

```

for e= 1 to E {set_counter(e); generate_initial_weight_system(w_old);}
for step= 1 to max_step {
  route_demands(w_old); for e= 1 to E cost_old(e)=compute_cost(e);
  if (network contains overloaded link(s)) for e= 1 to E {
  if(link e is underloaded) counter(e)=counter(e)-1;
  if(link e is overloaded) or ((link e is underloaded) and (counter(e)<0)) {
    cost_new(e)=compute_cost(e);
    w_new(e)=modify_weight(cost_new(e),cost_old(e),w_old(e));
    cost_old(e)=cost_new(e);
    w_old(e)=w_new(e);
    if(link e is underloaded) and (counter(e)<0) set_counter(e);
  }
}
if(all the links are underloaded) Load optimization method starts;

```

```

}
route_demands(w_new(e));
End

```

The algorithm starts from independent, random generated link weights w_e . If the network is overloaded the WA process will be started. All demands are routed according to the initial weight system, and the current cost of each link ($cost_new(e)$) is calculated as follows. If link e is underloaded, its cost is equal to its current load (occupied capacity) a_e , i.e. $cost_new(e) = a_e$. If the link is overloaded, its cost is a sum of two terms: one equal to the capacity y_e , and a second equal to the square of the link load minus its capacity, i.e. $cost_new(e) = y_e + (a_e - y_e)^2$ (the second term makes the algorithm reacts sensitively to the link overload). If a link is overloaded, its weight is increased in order to attempt to take away some demand flows from it. The magnitude of the increasing depends on the value of $cost_new(e)$ and the absolute value of the difference between the $cost_old(e)$ and $cost_new(e)$. If $cost_new(e)$ is bigger than $cost_old(e)$, it means that the overload on the link has increased in the previous iteration, the weight is increased by a value depending quadratically on the difference of the two cost values:

$$w_new(e) = w_old(e) + rand[1, 5] \cdot \frac{|cost_new(e) - cost_old(e)|^2}{cost_old(e)}$$

If $cost_new(e)$ is less than $cost_old(e)$, the overload has decreased, but the link is still overloaded. In this latter case the weight of the link is increased as well, but the value of the increase is smaller than in the former case:

$$w_new(e) = w_old(e) + rand[1, 3] \cdot \frac{|cost_new(e) - cost_old(e)|}{cost_new(e)}$$

When a link weight adjusted according to above formula exceeds the maximal weight (K), then it is set to K . The weight of an underloaded link is adjusted only at every several iterations. Our tests show that if we update the weights of underloaded links too frequently (e.g. in each iteration), disadvantageous cycles can occur, disturbing the convergence of the algorithm. Namely, most of the demands will be re-routed in each iteration and the links' loads will not fit their capacities since almost always there will exist a group of underloaded links, as well as a group of overloaded ones. To overcome this difficulty the following procedure is used. Each link e is assigned a randomly initialized attribute - $counter(e)$. If link e is underloaded, the value of $counter(e)$ is decremented at each iteration, and the link weight updated only if it becomes overloaded or if $counter(e)$ becomes negative. This assures that the algorithm will virtually avoid cycling, and additionally decreases the iteration execution time because of less link cost re-computations. When the link weight is adjusted due to the underload condition, $counter(e)$ is reset randomly. The random setting is another factor helping to avoid cycles in

the optimization. For an underloaded link the weight adjustment formula is:

$$w_{new}(e) = \left\lfloor \frac{w_{old}(e)}{\left(1 + \frac{cost_{new}(e)}{cost_{old}(e) + cost_{new}(e)}\right)} \right\rfloor$$

At each iteration the algorithm evaluates the current weight system with respect to (i) minimization of the number of overloaded links, and (ii) minimization of the magnitude of the average link overload. If the current solution is better than all the solutions found so far in any of these two aspects, the algorithm will store the current weight system. Hence, if during optimization the algorithm cannot find a feasible solution (i.e. a solution with all demands routed and no link overloads), the user can select between the two best solutions with respect to one of the two above criteria, and decide what is more advantageous: to minimize the number of overloaded links, or to minimize the average overload.

If the algorithm finds such weight system at which the network is underloaded the second part of the algorithm will be started. In this case the goal of the optimization is changed; the user can select from the three following alternatives:

- Maximize the average free capacity in the network
- Maximize the total free capacity in the network
- Minimize the variance of free capacity values on the links

According to the optimization goal several links will be selected (e. g. some most loaded and some least loaded) and their weights will be decreased or increased slightly, depending on the difference between the link's load (a_e) and the average load value ($\frac{\sum_{e \in E} a_e}{|E|}$) in the following way:

$$\text{If } a_e > \frac{\sum_{e \in E} a_e}{|E|} \text{ then } w_{new}(e) = w_{old}(e) + rand\left[1, 10 \cdot \frac{a_e - \frac{\sum_{e \in E} a_e}{|E|}}{y_e}\right]$$

$$\text{If } a_e \leq \frac{\sum_{e \in E} a_e}{|E|} \text{ then } w_{new}(e) = w_{old}(e) - rand\left[1, 5 \cdot \frac{\frac{\sum_{e \in E} a_e}{|E|} - a_e}{y_e}\right]$$

Then all demand will be routed according to the new weight system, the utilization and performance parameters of the network will be calculated. If the utilization of network is better (e.g. the algorithm moved towards the dedicated goal) the optimization is continued. If the utilization is worse then similar acceptance criteria to the simulated annealing method's will be used to decide whether the new state is acceptable or not. This acceptance method provides that the local optimum points can be avoided and the optimization time does not increase significantly comparing to simple greedy approach. If the current solution is better than all the solutions found so far, the algorithm will store the current weight system, therefore it can be used as a final solution at the end of whole optimization

process. It is a possible case that an acceptance of worse weight system results overloaded network. In this case the WA method will be used again. It seems that during the whole optimization time the weight adjustment and the load optimization processes work iteratively depending on the current network state (over- or underloaded network).

Summarizing the most important advantage of the SWA algorithm is the fast operation and the good solution, but sometimes problems are appeared, because of the cyclical and randomized operation.

3.5 Outline of the Weight Optimizer (WO) Method

On the basis of the experience with the SWA algorithm I decided to develop a new method called Weight Optimizer (WO), which is more complex than the SWA and can obtain better results.

3.5.1 Properties of the WO method

This section describes the most important properties, requirements and criterias of the WO method:

- Heuristic approach: It is evident that we cannot solve the optimization of real size networks using linear programming, but test results of SWA proved that the simple heuristics are not enough efficient to obtain good solution. Therefore I decided that the new algorithm will be based on combination of efficient and fast problem-specific heuristic subprocesses, but if it is possible I will take into account some results from the operational theory in the construction of procedures.
- One-phase approach: I decided to use the one-phase approach, because it seems to be more effective comparing with the two-phase one.
- Deterministic operation: I avoid stochastic, (random) heuristics and I work out an algorithm, which consists of only deterministic procedures. The goal was to prove, that a deterministic algorithm, which uses some considerations from the operational theory and contains problem specific procedures can obtain better results than the implementation of wide-known meta-heuristics (Simulated Annealing, Simulated Allocation, Evolutionary Algorithms) or simple iterative procedures.
- Robustness: The proposed algorithm consists of heuristic procedures. It raises the question, whether can we guarantee that the quality of solution does not decrease if the size of the problem increases or the network to be

optimized has special properties (e.g. special traffic pattern or topology). Our basic goal was to construct the algorithm so that its productivity is almost similar in cases of different size of networks.

- Scalability: It depends on an operator, the current optimization task and some other factors, what is more important: a short running time or a good solution. Therefore the method is scalable between very wide bounds to achieve the demanded equilibrium between the reasonable running time and quality of the result.
- The so-called *black box* type goal function was an obvious requirement, too. Because of this the algorithm is able to handle different (used-defined) network metric dependent goal functions.

3.5.2 Structure of the WO method

In this section I will summarize the basic ideas of the proposed algorithm and the task of subprocesses are overviewed.

During the optimization process, the network can be in two different states.

- Overloaded state (OLS): It means that at least the load of one link exceeds its capacity.
- Normal state (NS): It means that all links' loads are less equal to their capacities (no links are overloaded).

Accordingly, the algorithm has two different operation modes, OLS and NS mode. In OLS the goal of the optimization is to find such a weight system w , where there is no overloaded link in the network, while in NS the original goal is to maximize the residual capacity in the network or the black box goal function is used. The applied optimization processes in different states are designed for the two different goals.

In OLS mode the algorithm uses two kind of methods, and these try to reach a state, in which all links are underloaded; in other words, the processes try to move towards NS and reach a state, which is good enough for the further optimization, as fast as possible. The algorithm uses such goal function, which depends on both the number of overloaded links, and statistical parameters of the overload (average and variance of overload, localization of heavy loaded network segments). The operation of the methods is outlined in subsections 3.5.3 and 3.5.4, therefore we give only a brief general overview of them here:

- Set_link_weight procedure: It tries to re-route the traffic from the most overloaded links and equalize the utilization in the network.

- `Load_balancing_#1` procedure: It tries to locate and eliminate the overloaded parts of the network (where some overloaded links are connected to a common node).

If the algorithm arrives to NS, the user-defined or the original objective function is used. In this state three local search procedures try to find a solution close to the global optimum. Briefly, the task of the local search methods are as follows:

- `LS1: MAX-MIN_decreasing` procedure: It modifies the most and less loaded links' weights according to their load.
- `LS2: Variance_decreasing` procedure: It tries to decrease the variance of relative link loads, because tests show that this modification results better network utilization in most of the cases.
- `LS3: Load_balancing_#2` procedure: It tries to equalize the load between the different network parts.

The detailed operation of the methods are found in Sections 3.5.6, 3.5.7 and 3.5.8.

`LS1` and `LS2` do not use backtracking; if they move to a worse state it may happen, that some links will be overloaded, so the network will arrive in OLS again. In this case the OLS mode procedures will be used until the NS is achieved again. The network state specific procedures and the above iterative process between the two operation modes of the algorithm, depending on the current network state assures that a significant part of the whole state space will be examined during a relative short running time.

The methods used in OLS can start from any w system, while NS methods can start from such w system, which provides that the network will be in NS.

We should note that all of the above methods work independently from each other, each has its own task, but the adequate combination of them helps achieving good final solution in a great extent. It is also possible to skip some procedures from optimization process; this provides that the algorithm is scalable in a quite wide range.

The following subsections will give a detailed description of the applied procedures.

3.5.3 Set_link_weight procedure

This method is based on a pure iterative weight adjustment process using the links' current load with respect to their capacities as measure. The procedure works as follows.

- *Initial Step: Generation of a random initial vector w*
- *Step 1: Routing of demands using the current weight system w_{curr} according to the ECMP rule.*
- *Step2: The algorithm examines each link, calculates the difference between the allocated capacity a_e and y_e . Let $cap_e = y_e - a_e$ denote this difference for link e . If there is at least one link with $cap_e < 0$, the algorithm will go at Step3, else it will finish its operation, because the network is already in NS.*
- *Step3: The algorithm perform the following weight adjustment process link by link.*
 - *If $cap_e < 0$ and $w_e < K$, then $w_e = w_e + 1$.*
 - *If $cap_e < 0$ and $w_e = K$, then for each link $i \neq e$ and $w_i > 1$ $w_i = w_i - 1$.*
 - *If $cap_e > 0$ and $w_e > 1$ then $w_e = w_e - 1$.*
 - *If $cap_e > 0$ and $w_e = 1$ then for each link $i \neq e$ and $w_i < K$ $w_i = w_i + 1$*
- *Step4: Go back Step 1.*

The algorithm repeats the above procedure till a pre-defined step or while there is at least one overloaded link. If the procedure yields a weight system w which implies that the network is in NS, this w will be stored. Because this procedure modifies the weights of all links in each step, a significant part of the whole state space is examined. Another advantage of this method is its speed. In some cases the problem is that the effect of parallel weight adjustment is not predictable, therefore the convergence towards the NS solution is not continuous (the method can enter in cycle, but our experience is that usually the periods of cycles are more longer than the average step number of the method, therefore the cycles are not dangerous). To summarize, in spite of the simplicity of this procedure, it is able to find a weight system, for whom the network achieves NS in almost all cases.

3.5.4 Load_balancing_#1 procedure

The method of subsection 3.5.3 performs a link load based weight adjustment. Neither the state of the neighbouring links nor the overloaded parts of the network are considered during the optimization. This was a motivation to elaborate a method that takes a larger part of the network into account and provides more sophisticated weight calculation. This procedure is node-centralized, and in one iteration it tries to optimize the weights of those links, which are connected to a common node. The procedure takes into account the possible long-term effects of the weight changes in the current step. Let us introduce the following notations:

I_i set of incoming links to node i
 I_{i_ol} set of overloaded incoming links to node i
 O_i set of outgoing links from node i
 O_{i_ol} set of overloaded outgoing links from node i
 i_max index of the most loaded outgoing link from node i
 $cap_{i_max} = y_{i_max} - a_{i_max}$
 $sumcap_{i_in} = \sum_{O_i \setminus i_max} (y_i - a_i)$
 $sumcap_{i_out} = \sum_{O_i \setminus i_max} (y_i - a_i)$

The algorithm repeats the following procedure for all nodes $i = 0, 1, 2, \dots, N$. At first i_max is found at node i , then cap_{i_max} , $sumcap_{i_in}$, $sumcap_{i_out}$ are calculated. Let us see first the case, when $sumcap_{i_in} < 0$ and $sumcap_{i_out} + sumcap_{i_in} + cap_{i_max} < 0$. In this case all links in O_i and I_i will be examined. The weights of links are in I_{i_ol} will be increased by 2, for the sake of decreasing the overload of the incoming links. This is an important modification, because in many cases the outgoing links are not able to carry the flows of incoming demands and the demands generated by the current node. If – before the weight increasing – any $w_e = K - 1$, it remains the same, but such other link's weight will be reduced by 3, where $w_e > 6$. The weights are in $O_i \setminus O_{i_ol}$ subset will be decreased by one. The aim of this modification is trying to equalize the load between the outgoing links. If $w_e = 1$, it remains the same, but such other link's (in $O_i \setminus O_{i_ol}$) weight, at which $w_e < K - 3$ will be increased by 2.

We have to examine the case, when $sumcap_{i_in} > 0$, too. Then the algorithm differs three substates. In the first one $cap_{i_max} + sumcap_{i_out} < 0$ and $cap_{i_max} < 0$. This means that, although the incoming links have enough free capacity to serve the incoming flows, the outgoing set of links contains some overloaded links. To decrease their number the weights of links are in I_{i_ol} will be increased by 1. In the other subcase, when $cap_{i_max} + sumcap_{i_out} > 0$ and $cap_{i_max} < 0$, both incoming and outgoing links have enough free space to carry the flows, but on the outgoing side the load-balancing is not good enough. Therefore $w_{i_max} = w_{i_max} + 1$. It is important to note, that in both the above subcases the weight modification is permitted only if it is according to constraint (4), else the weight remains the same. In the third case, when $cap_{i_max} + sumcap_{i_out} > 0$ and $cap_{i_max} > 0$, the weights are not modified.

The algorithm repeats the above procedure during predefined step number, or while there is at least one overloaded link. In the last case the found weight system w will be stored, and this w will be the initial state of the NS optimization algorithms.

3.5.5 Methods in Normal State

The aim of the methods are presented in section 3.5.3 and 3.5.4 is to avoid the overloading of all links, by other words its task is to move the network from OLS to NS. In NS the original goal of the optimization is to maximize the free capacity in the network. That's why we need to use different kind of optimization strategies, than in previous two cases. Our experience was that a relative small change of the weight system can already result that the network goes into OLS again. Therefore, such methods are applied in NS, which do small, locally weight adjustment steps. It is important to note, that if during the optimization any of the next three methods found a state, which is better than all other found so far, the state is stored.

3.5.6 MAX-MIN_decreasing procedure

The goal of this method is to decrease the load difference between the links. That is why, in every step the most and the less loaded link is searched and their weights are increased and decreased by 1, respectively. Of course the weight modifications can happen only according to the weight related constraints. Because in this process there is no backtracking, it is possible, that after a modification the network leaves NS and become to OLS. Although, it is true, that the periodic jumping between the NS and the OLS has time-increasing consequences, but it has a great advantage, too. Namely, this jumping helps to examine much larger part of the whole optimization state space. This process is quite simple, hence its improvement is very small, but it can be used as a good measure. If this method cannot improve the solution, we found a quite good solution and the algorithm may be stopped.

3.5.7 Variance_decreasing procedure

The aim of this process is to minimize the variance of the load values in the network. The motivation was, that many numerical results showed, that the decreasing of load variance results the increasing of the total free capacity. In each step of the procedure the average free capacity value $cap_{aver} = \frac{\sum_{e \in E} (y_e - a_e)}{|E|}$ is computed. Then $a_e - cap_{aver}$ is calculated for each link e . If this value is larger than 0 (the load on the current link is larger than the average load in the network), then the weight of the link is incremented, while if less than 0, then its weight is decremented. After this weight setting, all demands are re-routed according to the new weight system and the cap_{aver} value is calculated again and the above process is repeated. Because cap_{aver} is calculated after every weight modification, this method can react to the smallest changes in the network. In the same way as in case of the previous method, backtracking is not used in this method, neither. (The operation of the procedure results that the backstep to OLS state is very rarely.)

3.5.8 Load_balancing_#2 procedure

Although the two previously described simple processes are working well, if we want to obtain near optimal solution we should use some kind of more sophisticated optimization procedure in NS, too. The tests showed, that the load balancing procedure in the OLS worked very efficiently, that is why I worked out another version of this kind of method for (the more effective) NS optimization. The aim of this load balancing procedure is the same as the previous one, but the used weight adjustment steps are different. Let us introduce the following notations:

$$\begin{aligned} cap_{var} & \quad \text{the variance of the free capacity} \\ cap_{var} & = \sum_{e \in E} (y_e - a_e)^2 - N \cdot cap_{aver} \end{aligned}$$

Initially, cap_{var} value of the current configuration is stored as cap_{var_prev} and the following procedure is repeated node by node. If for an incoming link of the current node i $\frac{|cap_e - cap_{aver}|}{cap_e} > 0.1$ condition is true, then the link is marked and put in a set denoted by I_{i_avdiff} . The outgoing links are examined in the same way and set O_{i_avdiff} is formulated. If $\frac{|I_{i_avdiff}|}{|I_i|} < \frac{|O_{i_avdiff}|}{|O_i|}$, then all outgoing links' weights are increased, if it is possible. Then the demands are routed according to the new weight system and the new cap_{var} value are calculated. If $cap_{var} > cap_{var_prev} + 0.8 \cdot cap_{var_prev}$ then the weight modification is not acceptable and the previous weights are re-adjusted. In case of the acceptance $cap_{var_prev} = cap_{var}$. If $\frac{|I_{i_avdiff}|}{|I_i|} > \frac{|O_{i_avdiff}|}{|O_i|}$, then all incoming links are examined and their weights are modified at the same way as described above. The acceptance criteria is the same, also. After the acceptance or rejection of the weight modification the method examines the next node.

As we can see this process uses more sophisticated method to select some weights to be modified. Because of this and the frequent demand re-routing this is the most time-consuming method, but it is able to search very fine way in the state space and it is very sensitive to the small changes, also. Hence this method is usable in any phase of the optimization in NS.

3.5.9 Framework of the WO algorithm

In the previous subsections 5 optimization processes were outlined, 2 for the OLS optimization and 3 for the NS optimization. As we have already mentioned any combination of those methods which works in the same network state is conceivable.

On Figure 3.1 the flow chart of the algorithm can be seen. On the right side of the chart the OLS methods, on the left side the NS methods are placed. In the first phase the aim of the optimization is to find a weight system, which allows NS in the network. When such w is found in any step the algorithm immediately finished the current operation phase and the NS methods are started. It is possible

to leave the NS again, then the OLS methods continue their operation.

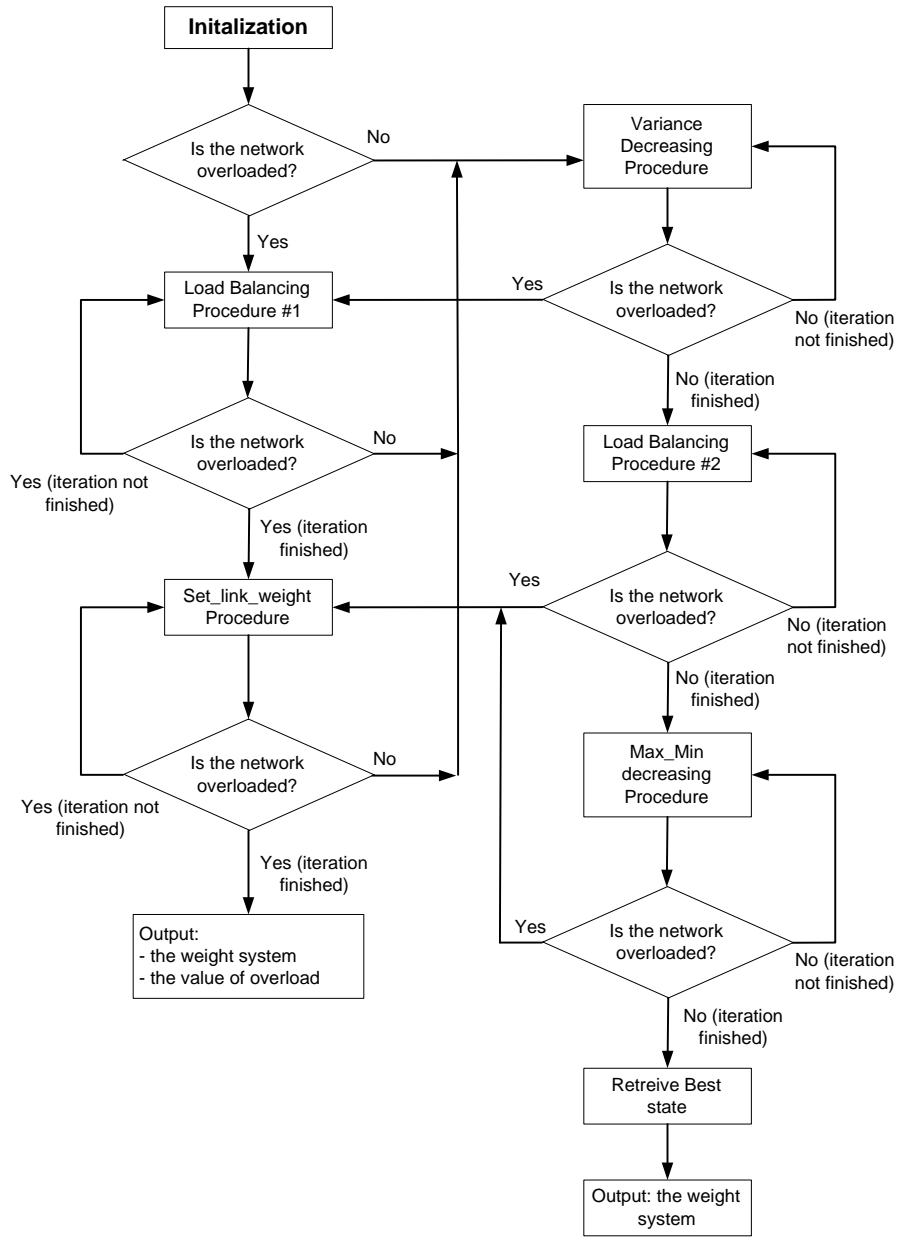


Figure 3.1: Flow Chart of the WO algorithm

The iteration steps of all procedures can be adjusted independently from 0 (skipping the current procedure from the optimization process) to any reasonable large number.

3.6 Performance Analysis

The aim of this section is to prove the efficiency of the proposed methods using different networks and test scenarios.

3.6.1 Test environment

Both algorithms are implemented in C++, and for all tests a SUN Ultra 1 (CPU: 143MHz, Memory: 128M) workstation was used. Table 3.1 shows the used test network configurations.

Table 3.1: Lower bound calculation

Network	N	E	D
N14_x	14	42	182
N28a_x	28	84	756
N28b_x	28	84	756
N56_x	56	224	3080
N112_0	112	672	12432

The topologies of the test networks are generated by a random graph generator algorithm. The values of demands are also generated randomly with truncated exponentially distribution between 120 and 5600. The link capacities of the networks were generated in the following way. A random w_{init} was generated and all demands were routed according to this w_{init} . Then - on the basis of the a_e values - y_e values were adjusted in the following way. If x (in N14_x) is 0 then the network is saturated, that means $y_e = a_e$ for all links. If x is 5 then the network is overdimensioned by averagely 5 percent; in case of all links, $y_e = 1.05 * a_e$, while if x is 10, then the overdimensioning ratio is 10 percent. The two 28 node networks have the same topology and demand number, but we used different demand volumes and the random generated w_{init} is also different. The saturated examples can be used for testing the algorithm in case of very high overall network load, while using overdimensioned network the algorithms can be tested in more realistic situations. The effect of the optimization (e.g. maximization of total free capacity) can also be measured using these networks.

3.6.2 Comparison test

In this section the proposed methods, SWA and WO, and two previously presented algorithms (denoted by SAN, SAL) are compared with each other. SAN is based on Simulated Annealing, while SAL used Simulated Allocation method, and proposed by Michal Pióro from Warsaw Technical University. For more detailed

description of the mentioned methods it is referred to [J5]. Since SWA, SAN and SAL uses randomized optimization more than one runs are required to test their performance. In cases of all the tests WA and SAL ran 100 times and SAN ran 50 times. Because of the deterministic operation, in case of WO one run is enough to obtain the best solution. The results are summarized in Tables 3.2, 3.3 and 3.4. The notation is as follows: AVRT - the average running time in seconds (the processor time is measured here), POPT - the percent of feasible solution found in all the runs. (We note that because WO is deterministic algorithm, consequently this value can be 100 or 0.) AVOL - the average number of overloaded links, AVTO - the average total network overload (the sum of exceeded links' load), and FNCR - the free network capacity ratio. We note that, in case of saturated networks FNCR is not explainable.

Table 3.2: Results of saturated networks

Net14_0	WO	SWA	SAN	SAL
AVRT[s]	1.03	0.03	~15 min	1.31
POPT[%]	100	100	100	100
AVOL	0	0	0	0
AVTO	0	0	0	0
Net28a_0	WO	SWA	SAN	SAL
AVRT[s]	0.57	0.2	~15 min	198.96
POPT[%]	100	48	100	0
AVOL	0	4.3	0	7.4
AVTO	0	720	0	9096
Net56_0	WO	SWA	SAN	SAL
AVRT[s]	169.57	155.8	~25 min	1352
POPT[%]	0	0	0	0
AVOL	41	61	18	85
AVTO	18923	63638.8	16912	24801
Net112_0	WO	SWA	SAN	SAL
AVRT[s]	561.9	243.4	-	-
POPT[%]	100	0	-	-
AVOL	0	81	-	-
AVTO	0	541548	-	-

In Table 3.2 the result of the saturated networks were summarized. The first and smallest example (Net_14) is very easy to solve. The difference between the algorithms can only measure in their running time. As we can see, SWA is the fastest, but WO can also find optimal solution during reasonable running time. Example Net28a_0 is already a more difficult task than the previous one. SAL

cannot find optimal solution ever, while SWA can find optimal solution only 48 percent of the runs. Although SAN is able to find optimal solution in all runs, but it requires a lots of iterations, consequently its running time is measurable in minutes. The third example is unsolvable (there is no such weight system, which provides that there is no overload). Using this example we can test how the algorithms work in heavy overloaded situations. We can see, that SAN can give the best solution, but WO's result is not much worse. The last network is quite large and we can examine the behaviour of our method when the optimization task for example a national backbone network. Neither SAN and SAL cannot give valuable solution during reasonable running time. Although SWA is able to solve the problem, but the quality of its final result is not very attractive. As we can see WO is also able to solve this problem. Consequently, in case of heavy loaded networks WO is able to find an optimal or a near solution during reasonable running time, independently from the size of the network and the difficulty of the optimization task.

Table 3.3 and 3.4 contains the results of 5% and 10% overdimensioned networks.

In case of Net14_5 all algorithms find the same solution, but – because of the more complicated procedures – WO can solve the problem during the longest running time. In case of Net28a_5 and Net28b_5 WO can find the optimal solution. As we can see Net56_5 is even a hard task. The 5% overdimensioning is not enough to find a NS solution. While in case of the saturated 56 nodes network SAL can give the best result, in this case we can see a very different situation. Result of SAL is very far from the optimum, and its running time is very long. SWA's results is also very poor. Result of SAN is acceptable, but we found its running time is about half an hour. In this example we can see the most important advantage of WO. It can find the best solution (what is almost NS state; less than 0.024% of the total traffic is lost), practically during the shortest running time, even in cases of large and heavy loaded networks.

The 10% overdimensioned networks are much more easier problems, than the previous ones, but I have few interesting comments on these networks, too. If you see any of the networks, you realize that WO presents the best solutions. Although the some tenth percent of free capacity increasing seems not too significant, we should note that already in case of the 28 node network it means quite great capacity value. It is also interesting that, in cases of greater network sizes, (where the state space of the optimization problem is greater) WO finds better solution than in case of smaller networks. A possible explanation of this behaviour is when the size of the network is small the algorithm find a NS local optimum solution in short time, consequently it cannot examine a great part of the whole state space in OLS to find a better starting point for the NS optimization.

On the long run, on the basis of the above tests, we may say that my proposed algorithms, SWA and WO can solve their task on an adequate way. SWA is able to

Table 3.3: Results of 5% overdimensioned networks

Net14_5	WO	SWA	SAN	SAL
AVRT[s]	17.96	3.31	~15 min	1.43
POPT[%]	100	100	100	100
AVOL	0	0	0	0
AVTO	0	0	0	0
FNCR[%]	4.8	4.8	4.8	4.8
Net28a_5	WO	SWA	SAN	SAL
AVRT[s]	68.29	19.36	~15 min	47.38
POPT[%]	100	100	100	100
AVOL	0	0	0	0
AVTO	0	0	0	0
FNCR[%]	5.00	4.9	4.8	4.9
Net28b_5	WO	SWA	SAN	SAL
AVRT[s]	40	19.36	-	-
POPT[%]	100	100	-	-
AVOL	0	0	-	-
AVTO	0	0	-	-
FNCR[%]	5.00	4.89	-	-
Net56_5	WO	SWA	SAN	SAL
AVRT[s]	156.53	155.77	~25 min	3211.2
POPT[%]	0	0	0	0
AVOL	3	12	4	33.3
AVTO	792	26658	2028	114411
FNCR[%]	0	0	0	0

find reasonable result during shortest running time, while WO is able to find better solutions than the another type of methods in almost all cases. The deterministic operation of WO provides the stable running time, and that one run is enough to achieve the best solution that the WO can find.

3.6.3 Effectiveness of NS state optimization of WO algorithm

In this section I will examine the effectiveness of the three different NS processes of WO, measuring the usefulness of them and I will note some scalability related comments. At first the percent of free capacity at the beginning and at the end of the current process is calculated. Table 3.5 shows the result of the examination.

At first we may say that all local search processes in NS is help to find a

Table 3.4: Results of 10% overdimensioned networks

Net14_10	WO	WA	SAN	SAL
AVRT[s]	12.39	3.19	~15 min	1.22
POPT[%]	100	100	100	100
AVOL	0	0	0	0
AVTO	0	0	0	0
FNCR[%]	9.2	9.1	9.1	9.1
Net28a_10	WO	WA	SAN	SAL
AVRT[s]	80.8	11.43	~15 min	16.95
POPT[%]	100	100	100	100
AVOL	0	0	0	0
AVTO	0	0	0	0
FNCR[%]	9.5	9.2	9.4	9.2
Net28b_10	WO	WA	SAN	SAL
AVRT[s]	29.14	20.41	-	-
POPT[%]	100	100	-	-
AVOL	0	0	-	-
AVTO	0	0	-	-
FNCR[%]	9.75	9.3	-	-
Net56_10	WO	WA	SAN	SAL
AVRT[s]	568.02	156.54	~25 min	2916.5
POPT[%]	100	10	100	30
AVOL	0	3	0	9.9
AVTO	0	2476.7	0	27458.4
FNCR[%]	9.64	-	9.5	9.1

good final solution. (But I have to note that improvement is happened during CYCLE_6 only in case of Net56_10, and the magnitude of improvement was not very significant – it can result some improvement only if we use another sequence of NS processes or we decrease the step number of the rest NS processes). Thus we may say that CYCLE_6 is not absolutely necessary, but on the other hand its running time is less than 4 - 5 percent of the total running time, consequently, we cannot achieve much faster operation if we skip CYCLE_6 from the optimization process. The other thing that I have to note is connected to the saving on running time. The first column of Table 3.5 contains the percent of free capacity at the first time the algorithm achieves NS. We can see that in case of Net14_5, Net14_10, Net28a_10, Net28b_10 and Net56_10, the value of free capacity is equal or greater than the final results of the other algorithms (SWA, SAN, SAL). It means that the OLS methods of WO usually can find same or better w than the final output of

Table 3.5: Effectiveness of NS state methods

Networks	FNCR in the first NS state w	FNCR after CYCLE_4	FNCR after CYCLE_5	FNCR after CYCLE_6 (final)
Net14_5	4.76	4.83	4.83	4.83
Net14_10	9.142	9.142	9.21	9.21
Net28a_5	4.73	4.84	5.0	5.0
Net28a_10	9.23	9.23	9.49	9.49
Net28b_5	4.78	5.0	5.0	5.0
Net28b_10	9.56	9.56	9.75	9.75
Net56_10	9.41	9.43	9.73	9.75

the other algorithms, and the NS methods of WO can further improve the quality of final solution. It means that if the running time is the most critical parameter it is possible to skip all NS procedures; the quality of the final results will be even good enough, but the running time saving is at least 30 percent comparing to the running time of the original configuration.

3.6.4 Issues on scalability of WO

Finally I would like to give some comments on the scalability. Two modes of scalability can be used in WO algorithm. In the first one the number of iterations of the NS and/or OLS cycles can be reduced, while in the other one, a whole NS cycle may be skipped.

If we choose the first way, the algorithm can be scalable in a wide range without any quality degradation of the final solution (iterations of NS cycles can be reduced up to 50), but if we decrease the iterations further, we will realize fast, but not critical reduction of the quality. The iteration number of OLS processes are also adjustable in a wide range, but if we set few step numbers we take a risk that the algorithm will have not enough iteration to find a NS compatible weight system. Therefore it is not offered to decrease the steps of OLS cycles significantly.

In the second case we can skip any NS procedures. To examine the effect of this scalability opportunity, we should take a look at Table 3.5. The results of different column suits to the skipping of different NS cycles. In the first column we can see the value of residual capacity if we skip the total NS optimization (all the cycles). Of course this results are much worse than with NS optimization, but we only can repeat that almost all of them are better than the final results of the other methods (SWA, SAN, SAL).

Of course the two way of scalability can be combined, but the further testing of this is out of the scope of my task to be solved. The results of this section and the previous one also shows that the algorithm is scalable in a wide range, because

neither the decreasing number of iterations nor the skipping of NS cycles does not result significant quality degradation.

3.7 Extension of the WO algorithm

In this section the possible extensions of the application of WO algorithm is presented. Two main questions seem to be interesting from the viewpoint of IP network optimization. The first one is how we can take already the effect of OSPF routing into account in the dimensioning phase of the network. (The dimensioning phase means the process when the capacities of the links are calculated on the basis of the offered or estimated (expected) traffic between node pairs.) The another question is: can we set the weight in the dimensioning phase such that if a link fault occurs then the traffic loss will be minimal? The first task may be referred as some kind of *network or link dimensioning* problem, while the second task is a *fault-tolerant network planning* type problem.

The WO algorithm is able to solve the first task without any major challenge in its operation. We can use the advantage of black-box type objective function, since if we define a new, adequate (typically step-like) cost function the algorithm is able to handle this kind of problem, too. The task here is to propose such a weight system, which provides the minimal sum of the capacity cost of the network. The weight system determines the bandwidth on the links, and on the basis of this value the required capacity for each link can be calculated. Because only pre-defined link capacities (e.g. OC-3, OC-12, OC-48, etc.) are available, we have to select the appropriate link capacities. The different link capacities have different costs according to the step-like link cost functions (in cases of simpler modeling linear or square-root type functions are used). Hence, the objective function of the optimization is to minimize the cost of the physical links, but no other modifications are required to obtain reasonable solution.

The second task is also solvable using WO, but here some minor changes are required in the optimization process. The detailed solution of the second task is out of the scope of this dissertation.

3.8 Conclusions

In this chapter an adequate network model and a MIP formulation of the static weight optimization problem in OSPF networks is formulated and two heuristic algorithms are proposed for solving the mentioned task. In case of the first algorithm (SWA) the most important factor was the short running time, while in case of the other method (WO) the deterministic operation and the best available solution were preferred. SWA uses two kinds of fast heuristic processes that are

able to find a reasonable solution during very short running time. WO algorithm consists of two main subprocesses, according to the two possible network states, namely there is any overloaded link in the network or not. In the first state, the goal of the optimization is to minimize the number of overloaded links and the value of overloading; while in the other one the algorithm tries to maximize the free capacity in the network. Both subprocesses contains several different local search procedures, which provides (a) great probability of finding a near optimal solution, (b) wide search in the state space, (c) reliable operation, independently from the different network topologies, traffic patterns. The test of the algorithms was very deep: its results were compared to the results of other known heuristics using some network sizes and traffic volumes. These tests proved that the proposed algorithms can perform the defined requirements. Namely, SWA can obtain reasonable solution during very short running time, while WO has a very stable, good quality operation even in cases of large networks and the final results are very close to the global optimum.

My further plan is to continue the research work towards two main lines. In the first case, the goal is testing the algorithm using step-like objective function, while the another goal is to integrate WDM related issues into our model and algorithm. In the latter case the algorithm will be able to solve the weight optimization task in IP over WDM or GMPLS/GMPλS networks.

Chapter 4

Dynamic Routing and Wavelength Assignment in Multifiber WDM Networks

4.1 Introduction

Nowadays, it is clear that WDM (Wavelength Division Multiplexing) networks will play fundamental and determinant role in the near future telecommunication world. Since WDM can provide an optical transmission system with extremely large data rate (currently 160 wavelengths per fiber is available, each with 10 Gbit/s bit rate), it is very attractive for long distance, high transmission speed backbone or core networks. In [22] it is mentioned that all major long-distance carriers in the U.S. have already used a point-to-point WDM transmission technology, and soon the *wavelength routing* will be introduced in most of the networks. The ultimate goal is the usage of so-called *all-optical* network, where a wavelength goes through the network without electronic/optical and optical/electronic conversion providing transparent data transmission.

Similarly to the introduction of the wavelength routed network (the point-to-point networks are replaced by wavelength routed networks), the dynamic establishing of traffic demands also becomes more and more important. Assumption of dynamic traffic arrivals can be explained by the following reasons. First, the continuously increasing traffic volume and the so-called "bandwidth hungry" applications yield that the offered traffic of Internet service providers will be very high. Furthermore, the demand for this kinds of service will change from time to time since the demands of the customers are changing as well (the time scale can vary between minutes to hours). Second, it is possible to reconfigure the network in response to changing traffic patterns or link/node failures (in this case the time frame is larger).

This chapter deals with the emerging on-line (on-demand) wavelength selection and wavelength routing (**WS & WR**) problems in WDM networks with dynamic traffic arrival model. My objective is to propose an algorithm that is able to solve the dynamic connection establishment problem (routing and wavelength assignment) in an optimal way, where the network performance is measured in the blocking probability. In addition to the presentation of our proposed allocation strategies my another main objective is to give a detailed analysis of them, applying different network configurations and traffic scenarios. Nowadays, this subject is studied extensively in the literature. In the following we give a brief overview of published efforts and results in this area.

The above mentioned problem was first discussed in [23], where the authors assume fixed routing (the routes are given in advance) and a greedy heuristic, called *First-Fit* algorithm for solving the WS problem. In this case the wavelengths are assumed to be indexed arbitrarily and the new connection is established on the available wavelength that has the smallest index.

In [24] the authors proposed a layered graph model of the WDM network and developed two wavelength selection methods, called *Pack* and *Spread*, furthermore, they presented a faster version of Dijkstra's method.

In [25] Zhang and Qiao proposed an on-line wavelength assignment algorithm for multifiber WDM networks. They define relative capacity loss for each wavelength and choose the wavelengths with minimum relative capacity loss. For a given number of fibers per link and number of wavelengths per fiber, the algorithm aims to minimize the blocking probability. However, for each potential connection, fixed routing is used.

In [26] the First-Fit algorithm was compared to the *Random Wavelength Allocation* and simulations show that the blocking probability of the first fit algorithm with fixed routing is considerably lower than the blocking probability with the Random Wavelength Selection algorithm. Furthermore, the authors propose the so-called *Least Loaded Routing* algorithm for mesh type networks with and without wavelength conversion.

In [27] the idea of packing wavelengths is further extended in the *Most-Used* algorithm where the highest loaded and still available wavelength is selected. The authors adopt a general adaptive routing approach, where all paths are considered between a source destination pair. The authors propose some mechanics for the wavelength search sequence and evaluate their blocking performance using both single- and multifiber networks.

In [28] it is shown that the Most-Used algorithm performs slightly better than the First-Fit one for ring topology. Besides, an algorithm is proposed where the wavelength is selected that maximizes the total network capacity after the connection is established. The authors pointed out that their new algorithm can also be extended to alternate routing using k shortest paths between sources and sinks.

In [29] the authors calculated the approximate blocking probabilities for fixed routing, while in [30] this technique has been extended to alternate routing with random wavelength allocation.

In [31] dynamic routing algorithms based on path and neighborhood link congestion are proposed assuming fixed path.

In [32] the authors propose some sophisticated, link state dependent weight functions for the Dijkstra algorithm and they evaluate the performance of them.

For more details we refer to [33] that contains a detailed overview of the activities on WDM transport network related questions.

On the basis of the literature survey, I have some comments on these papers. At first, one group of the previously presented algorithms use fixed routing, or alternate routing, which means that the available path(s) for each demand are pre-defined, so the main problem is to find an appropriate wavelength for the connection. This yields suboptimal solution since the blocking probability is lower with dynamic route selection. Secondly, the wavelength selection and the routing problem were solved in one phase, using some layered-extension of the network graph as it is presented in [24]. Furthermore, other group of the papers concentrate on finding an adequate path for a demand using a sophisticated Dijkstra's function, but the WS problem is solved in a very simple way, namely using random (RAND) or simple sorted wavelength selection (SORT) processes.

The main focus of my research work was to fill these gaps of existing algorithms. I propose a resource allocation method, which handles the WS and WR problems in integrated way using the current network status information. Since the wavelength selection and routing problem is known to be \mathcal{NP} -hard [33], heuristic is necessary. For the WS problem my proposed a method is called EXHAUST (outlined in Section 4.3.4). In case of EXHAUST all wavelengths are examined according to the pre-defined network state metric(s) (e.g. load of the current wavelength, path length on the current wavelength using an adequate weight function) and the best one is selected, which provides lower blocking probability in long time frame. For the comparison I implement SORT and RAND methods based on [26]. For the WR problem I propose different several sophisticated link weight functions that consider the current load of the links and wavelengths (outlined in Section 4.3.2). For comparison the simple shortest path routing is used.

I consider "all-optical" two-connected core or backbone networks with Optical Cross Connects (OXC) and Optical Add-Drop Multiplexers (OADM) in each node. It means that all nodes have switching capability, and any node is able to generate a demand towards any other one. Furthermore, I assume transmitters and receivers are tunable in all OADMs, which means a demand can be established on any available wavelength.

The nodes are assumed not to have wavelength conversion capability. It is evident that the blocking probability of this kinds of network is larger than the

network with wavelength conversion, but in [34] it is shown that the difference is very small. If we can pay the small cost of a little bit higher blocking probability, we can save the cost of very expensive wavelength converters. In summary, the cost of the network without wavelength conversion capability is much less than with wavelength conversion, and the performance degradation is absolutely not significant. Furthermore, the links are assumed multifibers that means several fibers are installed between the nodes and each fiber contains the same number of wavelengths.

4.2 The Network Model

In a general case we can model any optical network by a directed graph $G(\mathcal{N}, \mathcal{L}, \Lambda_Q)$, where \mathcal{N} represents the set of nodes, \mathcal{L} represents the set of links, and Λ represents the available wavelengths in the network, while Q denotes the number of different wavelengths. Since we assume multifiber network, an integer $C(l)$ is defined for all links $l \in \mathcal{L}$, representing the number of fibers installed in this physical connection. Summarizing, our network representation requires unique wavelength number (Q) per fiber, but enables different number of fibers on different links. Furthermore, we assume the demands are also directed. The reason of directed physical links is that the fiber number can be different in the opposite directions.

In the algorithm, instead of unique G , a set of graphs $g_{\lambda(i)}(\mathcal{N}, \mathcal{L}, \lambda)$ is used, where $\lambda(i)$ is the i^{th} wavelength, $i = 1 \dots Q$. Each $g_{\lambda(i)}$ graph represents the available network resources on the adequate wavelength $\lambda(i)$. For each $g_{\lambda(i)}$ graph, for each link l a so-called state (column) vector $\chi(i, l) = (x(i, l, 1), x(i, l, 2), \dots, x(i, l, z), \dots, x(i, l, C(l)))$ is defined. If $x(i, l, z) = 1$, then fiber z of link l in graph i is occupied, otherwise $x(i, l, z) = 0$.

From the viewpoint of dynamic arriving of connection establishments (and tear-downs), I consider a discrete system, i.e., the demand establishing requests or demand tear down requests arrive in arbitrary times $t_1 < t_2 < \dots < t_k < t_{k+1} < \dots < t_K$ (these events are called *actions* in the following). Only one action is assumed at a time t_k . For any time t_k we may introduce the step-dependent version of state vector $\chi(i, l)$ using the following notations: $\chi^k(i, l) = (x^k(i, l, 1), x^k(i, l, 2), \dots, x^k(i, l, z), \dots, x^k(i, l, C(l)))$. This vector shows the load of the adequate $\lambda(i)$ in step k . I have to note that the condition of the appropriate operation of the algorithm is that $\min_k(t_{k+1} - t_k)$ must be greater than the required path computation time; it is assumed that this condition is fulfilled in the network and simulation model.

Some network performance metrics are introduced, which will be used in the WS strategies. Let N_a^k mean all of the allocated and N_b^k mean the number of blocked demands *until* step k . Based on the above the average blocking probability until step k is denoted and computed in the following way: $\mathcal{P}_b^k = \frac{N_b^k}{N_a^k + N_b^k}$.

Let $\omega_{\lambda(i)}(k)$ denote the number of currently allocated demands on wavelength $\lambda(i)$ in step k . Using this $N_a(k) = \sum_{i=0}^Q \omega_{\lambda(i)}(k)$. We define the load of a wavelength in step k by $\psi_{\lambda(i)}(k) = \frac{\sum_{l=0}^L \sum_{z=0}^{C(l)} x^k(i,l,z)}{\sum_{l=0}^L C(l)}$.

4.3 The Proposed Wavelength Selection and Routing Methods

In this section we outline the assumptions for the algorithm and the network environment, we overview the used weight functions of the routing algorithms and the wavelength allocation strategies, finally we give a brief pseudo-code of the proposed WS&WR algorithm.

4.3.1 Assumptions for the Proposed Algorithm and Network Environment

Before outlining the proposed methods, it is important to clarify the assumptions, condition and constraints for the network environment and the WS&WR algorithm.

1. The topology of the network is known and does not change during the working period of the algorithm (except in case of link failures).
2. The source and sink nodes of the incoming demands are known, while the holding time (till the connection will be alive) is not, consequently this information is not usable during the WS&WR phase.
3. There is no information about future incoming requests. It means that the algorithm can only take the actual network status into account.
4. We assume that a demand can reserve one wavelength.
5. The algorithm has information on the current status of the network (e.g. the paths of the established demands). Normally the algorithm operates as a centralized tool, but it is also able to work in the OXCs in a distributed way. In the first case the demand establish or tear down requests are forwarded to the queue of a central management unit. Using this model, the requests are processed in a FIFO mode and it is obvious that the central management tool has recent and precise information about the state of the network. If the WS&WR problem of the current demand is solved, the management tool can handle the required modifications in the adequate nodes and links to establish or tear down the demand. Consequently, our algorithm is suitable to operate in a centralized mode without any restriction.

In case of distributed operation we can realize some difficulties. First of all, the most important problem is the propagation delay of the link state advertisement messages. If $\min_k(t_{k+1} - t_k)$ is smaller than the required time for all nodes to get information about the latest network state then some nodes have not up to date information about the used network resources, resulting faulted wavelength selection and route computation causing demand blocking in the resource allocation phase. We can eliminate the problem if we provide that $\min_k(t_{k+1} - t_k)$ is larger than the sum of the computation time of the algorithm and the largest propagation delay. Summarizing, the distributed operation is not impossible, but we have to consider that some demands can be blocked in the establishing phase.

4.3.2 The Proposed Weight Functions

During the wavelength selection (Subsections 4.3.3 and 4.3.4) we have to solve the routing problem of the incoming demands. In our model the WS and WR problems are strongly connected to each other, since the wavelength selection is influenced by the applied weight function of the routing algorithm (see $w_{func(l)}$ in the pseudo-codes in the next sections).

Eight different types of weight functions are applied here. The first function is widely-used, but it does not take the link load into account, the central links of the network will be overloaded causing high blocking probability. Therefore I propose seven weight functions, which are able to react to the current status of the network and try to adaptively share the load among the links. The functions are listed below:

Reference function:

- (1) $w_{func(l)} = 1$ for all links. This is the simple fixed, shortest path routing, which is the most-used function in the known wavelength selection methods. The function absolutely does not consider the number of used fibers, therefore the load balance in the network is often very far from the optimal, which causes great blocking probability.

Proposed new functions:

- (2) $w_{func(l)} = \sum_{z=0}^{C(l)} x(i, l, z)$ in wavelength $\lambda(i)$. Using this function the demand is routed onto those links, on which the number of used fibers is the smallest.
- (3) $w_{func(l)} = \frac{1}{C(l) \cdot (C(l) - \sum_{z=0}^{C(l)} x(i, l, z)) + \varepsilon}$ in wavelength $\lambda(i)$; ε is a small number (e.g. 10^{-4}) and it is required to avoid the division by zero.

- (4) $w_{func(l)} = \frac{1}{\left(\frac{C(l) - \sum_{z=0}^{C(l)} x(i,l,z)}{C(l)} + \varepsilon\right)}$. This function is very similar type to the previous one, but it react more sensitively if the load
- (5) $w_{func(l)} = \frac{1}{\left(C(l) - \sum_{z=0}^{C(l)} x(i,l,z)\right) + \varepsilon}$. This is a little bit simplified function, but in spite of this it seems to be quite effective in case of heavy loaded networks.

In case of functions (3) (4) and (5) the weight of a heavy loaded links are very large (the the denominator of the formula is close to zero when the link is heavy loaded), therefore these links will not be loaded further if there is at least one less loaded path for the demand.

- (6) $w_{func(l)} = \frac{\sum_{z=0}^{C(l)} x(i,l,z)}{(C(l)+1)}$. In this case also the least loaded path is selected, but the weight of the link is normalized.
- (7) $w_{func(l)} = \frac{\sum_{z=0}^{C(l)} x(i,l,z)}{C(l)}$. This function increases linearly with the load of the link, which may cause that the difference between the load of the wavelengths can be significant.

Although the above 6 functions help to select the least loaded path and we can achieve very good load balance between the wavelengths, but the paths would rarely be longer than the shortest paths (measured in hops), causing some non-desirable extra load in the network.

- (8) $w_{func(l)} = \delta + \frac{\sum_{z=0}^{C(l)} x(i,l,z)}{C(l)}$. This weight system considers a correction part ($\delta \geq 1$), which forces the routing algorithm to select a shorter, more loaded path if the least loaded path is too long. The function helps also to select the least loaded between the equal-length shortest paths.

4.3.3 Reference (Existing) Wavelength Selection Strategies

In this subsection the two well-known wavelength selection methods (SORT and RAND) are described. Both of these methods are based on first-fit wavelength selection strategy, which provides simple and relative fast operation, but the cost of them is the performance degradation of the methods.

- **SORT** strategy: Using this type of wavelength allocation each wavelength is examined in a pre-defined order and the first one is selected, where the demand allocation is possible.

```

FOR i=1 to Q DO {
  routable = CheckDemand( $a, g_{\lambda(i)}, w_{func(l)}$ )
  IF (routable == "YES") {
    AllocDemand( $a, g_{\lambda(i)}, w_{func(l)}$ )
    DEMAND ALLOCATION IS POSSIBLE; Exit loop;
  } end IF
} end FOR
IF (routable == "NO") THEN BLOCKING!

```

Function $\text{CheckDemand}(a, g_{\lambda(i)}, w_{func(l)})$ tries to allocate demand a into wavelength $\lambda(i)$ using weight function $w_{func(l)}$. If the allocation attempt was successful, the demand is allocated on the current wavelength (indexed by i) on the shortest path determined by $w_{func(l)}$ (AllocDemand). If all wavelengths have already checked and there is not enough free resource for the allocation then the demand request is blocked. The advantage of this greedy method is its simplicity and fast operation (the demand is allocated on the first suitable wavelength), but its drawback derives from the greedy property (the first fit wavelength does not mean the best fit wavelength). A sophisticated searching and selection between the wavelengths which are able to carry the new demand can result much better long-term network performance and load balancing (using SORT, the wavelengths with smaller index are higher loaded, while others lower).

- **RANDOM** strategy: In this case a wavelength is selected randomly and it is checked whether the demand can be allocated on it or not.

```

FOR j=1 to Q DO {
  i = RANDOM(Q)
  routable = CheckDemand( $a, g_{\lambda(i)}, w_{func(l)}$ )
  IF (routable == "YES") {
    AllocDemand( $a, g_{\lambda(i)}, w_{func(l)}$ )
    DEMAND ALLOCATION IS POSSIBLE; Exit loop;
  } end IF
} end FOR
IF (routable == "NO") THEN BLOCKING!

```

The WS process in this case is very similar to the previous one, the difference is that the sorting of examined wavelengths is different (random) for each demand request. There are Q attempts to find an available wavelength for the new demand, so each wavelength is examined only once, but the sorting of wavelengths is different for the demands. In long term this method requires about the same computation time as SORT strategy, but because of the

randomization, the load balancing between the wavelengths is a more uniform than in case of SORT method.

4.3.4 The Proposed Wavelength Selection Strategy

EXHAUST strategy is the collection of sophisticated wavelength selection methods, which are able to perform better overall network utilization and lower blocking probability than the above wide-used processes. Using any version of this method all wavelengths are examined and the best one is selected according to some pre-defined network measures (see below).

```

best_wl=-1
FOR i=1 to Q DO {
  routable = CheckDemand( $a, g_{\lambda(i)}, w_{func(i)}$ )
  IF (routable == "YES") {
    best_wl=Comparison( $i, best\_wl, a, w_{func(i)}, WS\_mode$ )
  } end IF
} end FOR
IF (best_wl != -1) {
  AllocDemand( $a, best\_wl, w_{func(i)}$ )
  DEMAND ALLOCATION IS POSSIBLE;
} end IF
IF (best_wl == -1) THEN BLOCKING!

```

As we can see, the WS process is more complicated than in the previous two cases. All wavelengths (indexed by i) are examined one by one, and the most suitable one (this is $best_wl$) is selected based on WS_mode for demand allocation. The comparison is done using the following performance measures, defined by WS_mode attribute.

- If $WS_mode = 0$, then that wavelength is selected first, where the demand can be realized on the shortest path according to w_{func} . If there are more than one such wavelengths, then in the next phase, from these wavelengths that one is selected, on which $\omega_{\lambda(i)}$ is the smallest one. This process provides that always the less loaded wavelength is selected among those on which the paths are shortest, consequently the load in the network is kept in equilibrium between the wavelengths.
- If $WS_mode = 1$, the first phase of selection is the same, but in the second phase that wavelength is selected, where $\omega_{\lambda(i)}$ is the largest one. In this case always the most loaded wavelength is selected among those on which the paths are shortest, resulting maximum utilization of the available wavelengths. Although this strategy does not provide as uniform load distribution

between the different wavelengths as the previous one, but it is expected that later a new demand can be established in one of the less loaded wavelengths with larger chance.

- If $WS_mode = 2$, the first phase is also the same as in $WS_mode = 0$, but in the second phase that wavelength is selected, where $\psi_{\lambda(i)}$ is the smallest one. In this case the load of a wavelength is measured better than in the previous cases because in addition to the number of allocated demands ($\omega_{\lambda(i)}$), the length of the paths are also considered in the computation of the load ($\psi_{\lambda(i)}$).
- If $WS_mode = 3$, the first phase is also the same as in $WS_mode = 0$, but in the second one that wavelength is selected, where $\psi_{\lambda(i)}$ is the largest one.
- If $WS_mode = 4$, then that wavelength is selected, where $\psi_{\lambda(i)}$ is the smallest (the wavelength is the less loaded) independently from the length of the path of the demand to be established. In this case the least loaded path is selected, but it can cause some additional load increasing, if the path of the demand is too long.

In the further part of the chapter we will refer the different setting of WS_mode attribute as $EXHAUST(WS_mode)$; for example $EXHAUST(0)$ means $WS_mode = 0$ strategy.

We would like take some notes on running time of $EXHAUST$ method compared to $SORT$ and $RAND$. If we apply $EXHAUST$, all the wavelengths are examined, which requires Q iterations. In long term, in case of a less loaded network, the average required iterations of $SORT$ and $RAND$ is $\frac{Q}{2}$. Otherwise, if the network load is high the number of required iterations is very close to Q in case of both $SORT$ and $RAND$. It causes that in a network where the average load is near 70-80% there is no significant running time increasing if we use the $EXHAUST$ instead of simple $SORT$ and $RAND$ methods.

4.3.5 Issues on fairness between demands

Most of the on-line allocation algorithms consider only the long-term, overall blocking probability in the network as the basic measure of the goodness of the proposed allocation strategy. This kind of evaluation can give a good view about the performance of the allocation method's effectiveness on network level, but we have no any information about the individual blocking probabilities (denoted by p_b^i for user i) user by user. Therefore it is possible that the user level blocking probability can be very different from the overall network blocking probability (P_b). By other words the fairness between the users is not provided, because some of them see (significantly) greater, others see lower blocking probability than P_b . It results that a relative advantageous value of P_b is not enough for the satisfaction of

all users; for them $p_b^i \gg P_b$. To eliminate this, my proposed allocation strategy is able to handle the fairness requirement between the demands, if it is an important condition of the network operator.

If the algorithm considers the fairness criteria then during the network operation the p_b^i values for all users are recorded and updated after each demand allocation attempt. In case of the allocation of a new demand of user i the p_b^i value is considered in the following way. If $P_b - p_b^i > \Delta$, then the demand is blocked even if the allocation is possible (δ can be adjusted by the network operator). This process provides indirect way that there will be more free resource in the network for that users who have worse p_b^i value. If we examine the network in long-term then all p_b^i value will be close to each other, which means that the fairness criteria is fulfilled between the users.

4.3.6 Path length examination

In some heavy loaded network cases it is possible that a demand use much more resources than in a less loaded situation. A typical example for this case is when a demand use a long roundabout path, which contains significantly more links than a path in regular network state. If we enable the allocation of a demand, which uses roundabout path it can occur that some other demands are blocked because of the unefficient resource sharing. If the network remains in heavy loaded state in the long-term this process can results that the overall blocking probability increases, because the network contains many roundabout paths that uses more resources than it would be required. My proposed length examination method helps to decrease the effect of roundabout paths. Before all demand allocation, the current length of the path (measured in number of hops (used fibers), denoted by d_a^l) is compared to the length of those path which would be used by the demand in an empty network (denoted by d_a^e). Then the following inequality is evaluated: $(d_a^l - d_a^e) \geq \phi(n_v, n_w)$, where $\phi(n_v, n_w)$ is also measured in hops, and can be specified independently between any node pair. If the inequality is true then the demand will be rejected even if there would be space to allocate it. Consequently, the process filters the quite long roundabout paths, so the long-term operation of the network will be more efficient and even in heavy loaded situation the number of serviced demands can be increased.

4.3.7 Pseudo-Code of the Algorithm Frame

In summary, the user can select from two main types of existing wavelength allocation strategies (SORT, RAND) and our proposed method (EXHAUST). Within the EXHAUST, 5 selection strategies may be selected. It means that 7 different strategies may be selected and they are combined with 8 weight functions, resulting

56 different configuration possibilities of the WS&WR algorithm. The following pseudo-code helps to understand the operation of the algorithm:

```

DEFINE WL_Sel_Strat (1-7)
DEFINE Weight_Type (1-8)

IF(ACTION)  CHECK(ACTION)
  IF(ACTION = Demand  $a$  Setup Request) {
    FLAG = WS&WR PROCESS( $a$ , WL_Sel_Strat, Weight_Type)
    IF (Fairness examination activated) {
      Fairness_Examination_Process()
      IF (Fairness is broken) FLAG = FALSE
    }
    IF (Path length examination activated) {
      Path-length_Examination_Process()
      IF (Path-length criteria is broken) FLAG = FALSE
    }
    IF (FLAG = SUCCESS) RESOURCE ALLOCATION( $a$ , Sel_Path, Sel_WL)
    IF (FLAG = FALSE) Demand Request Refused
  }
  IF(ACTION = Demand  $a$  Tear Down Request) {
    RESOURCE FREEING( $a$ )
  }
}

```

Before the beginning of the operation the user sets the selected WS and WR strategies. The algorithm needs the following information about the network: the nodes, the physical links connected to them and the number of wavelengths (this is graph G), as well as the number of fibers per links ($C(l)$ for each link l). From these data the algorithm will build the $g_{\lambda(i)}$ graphs.

As we have mentioned, in our context *action* means a demand establishment request or a demand tear down request. If an *action* is recognized, the algorithm first checks the type of it. If it is a demand setup request then a Wavelength Selection Process starts (Subsections 4.3.3 and 4.3.4) according to the pre-defined selection strategy and weight function type (WL_Sel_Strat, Weight_Type). If this process was successful (Flag=1), the required modifications are executed in the adequate $g_{\lambda(i)}$ graph of the algorithm and the data of the computed route (Sel_Path, Sel_WL; the selected wavelength and path) are forwarded to the network management to reserve required resources for the demand in the real network. If the computation was unsuccessful (Flag=0) then the demand request is rejected. If the *action* is demand tear down request then the only task is to release the resources in the adequate $g_{\lambda(i)}$ graph.

4.4 Performance Analysis

We have conducted numerical investigations in order to show the real improvements resulting from using the proposed WS&WR algorithms. In Section 4.4.1 the simulation environment and methodology is described while in the Section 4.4.2 the simulation experiments on blocking probability will be presented and analyzed.

4.4.1 Simulation Environment

We used three networks for simulations: the topology of $Net_{AT\&T}$ is the same as the national backbone network of AT&T in the USA, Net_{Pl} is the national backbone network of Poland, while Net_{56} is a random generated network with a dense mesh topology. For simplicity reasons, we assume that the number of fibers per link ($C(l)$) is equal for all links. Table 4.1 contains the most important data of the networks (N-number of nodes; E-number of edges; M-number of demands; C-number of fibers per links; Q-number of wavelengths).

Table 4.1: The test networks

Network	N	E	M	C	Q
$Net_{AT\&T}$	25	88	1800	4	8
Net_{Pl}	12	36	660	5	8
Net_{56}	56	224	3080	6	8

For the description of the demand allocation/deallocation request process we have some mathematical tools in our hand. The most widespread demand arrival model is Poisson-process where the requests arrive with a given α average intensity or the interarrival time between two requests is exponentially distributed with parameter τ . Another possible model is when the requests arrive at equal time intervals while the number of active intervals is distributed exponentially. Although the above models are suitable for our problem, we applied another demand request arrival model instead of them, which helps to describe and compute the network performance parameters better.

We use a list of actions (referred to as `action_list`), which contains information about the demand request actions and is generated using the following methodology. At first, we have to define the number of actions during the simulation, denoted by N_{act} . Then the pre-defined average load of the network (L_{goal}) to be achieved, and the variance of this value (V_{goal}) is to be specified. In our context L_{goal} means about how many percent of the demands should be active in the network during a long time frame. (Please consider that in some cases L_{goal} can be only a theoretical load because it is possible that the network cannot carry this

amount of demands that can result this "set aside" load.) Then an initial process is started, in which about as many demand requests are generated that provide load L_{goal} in the network. Let this current load value denoted by L_{curr} (in the simplest case as many demand setup requests are generated which provides that L_{curr} is equal to L_{goal}). After this initial phase the following process is repeated N_{act} times:

- If $|L_{curr} - L_{goal}| \leq V_{goal}$, a demand will be selected randomly using uniform distribution, and if currently it is active then it will be deallocated, while if it is not established then a demand setup request will be generated.
- If $L_{curr} - L_{goal} > V_{goal}$, then it means that the network seems to become overloaded compared to L_{goal} therefore a demand will be deallocated.
- If $L_{goal} - L_{curr} > V_{goal}$, then it means that the network seems to become underloaded compared to L_{goal} therefore a demand setup request will be generated.

This generation method provides that the average network utilization will be between well-definable bounds, and the test pattern will not contain uncontrolled parts where the average load or demand request intensity changes locally. On the other hand, using V_{goal} we can set an interval where the demand setup and tear down requests are generated randomly with independent, identical distribution. The aforementioned `action.list` is used as an input file of the algorithm during the different simulation scenarios.

4.4.2 Simulations on Blocking Probability

In this section we investigate the performance of the WS&WR algorithm using different adjustment of wavelength selection strategies and weight functions.

First of all, we would like to prove that sophisticated WS strategies (the different settings of the EXHAUST method) can provide better long-term, overall network performance than the widely applied methods, SORT and RAND. In the simulation all of the test networks are used to examine how the algorithm performs in different network sizes. In our simulations for $Net_{AT\&T}$ and Net_{Pl} 15000, while for Net_{56} 20000 demand requests are generated to ensure that steady state has been reached (it means that the traffic has a fixed intensity and the blocking probability goes to an average value in the long term). In this test all possible settings of the WS methods are examined, but for simplicity's sake (to reduce the number of test scenarios) we applied only weight function type 3. In the first test we started from such a demand request arrival intensity which causes that about 50% of the demands are active at a time, then we increase the average number of active demands by 5% test by test until it reaches 95%. During the simulations

the fairness and path length examination processes are not used. The results are summarized in Figure 4.1, for Net_{PI} , $Net_{AT\&T}$ and Net_{56} , respectively.

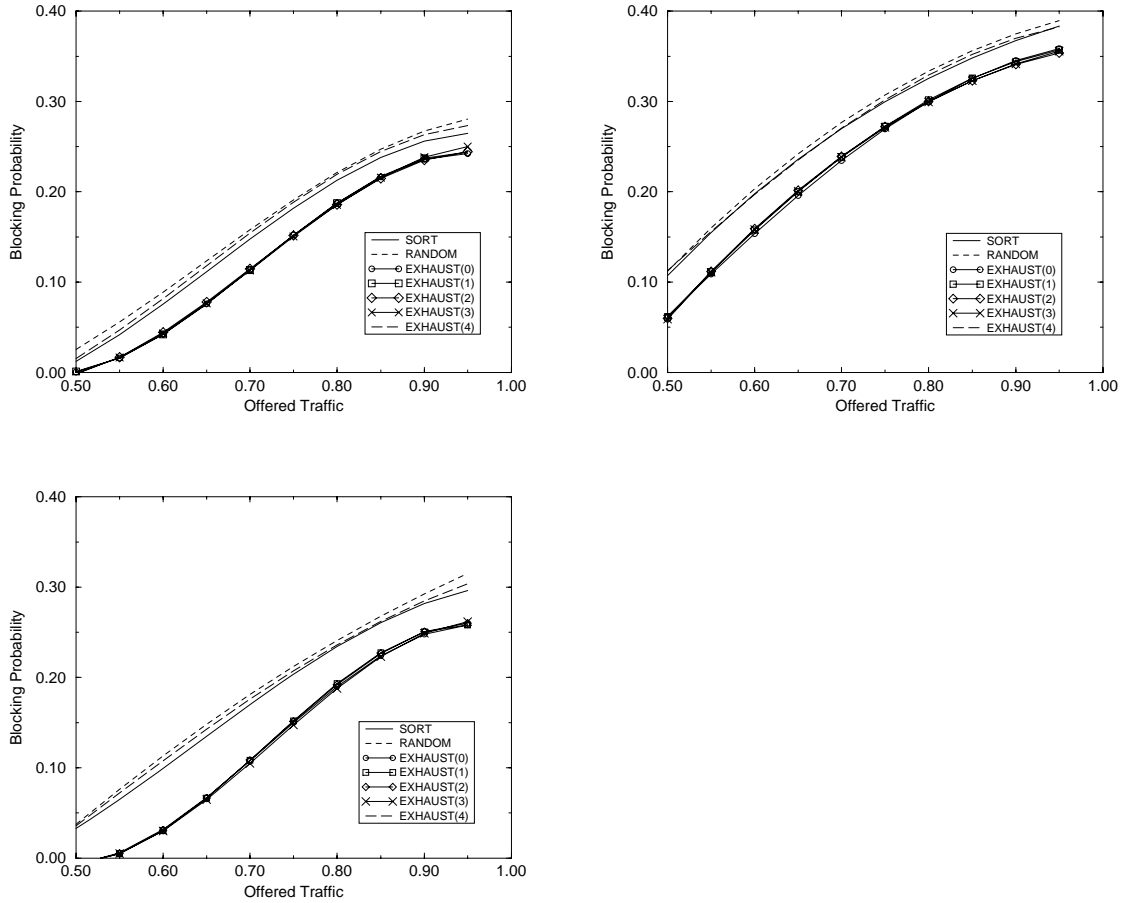


Figure 4.1: Blocking probability using different WS strategies in Net_{PI} , $Net_{AT\&T}$ and Net_{56}

Considering the figures, in all of them two main groups of blocking curves are sharply distinguished and the groups contain the curves of the same wavelength selection strategies in all networks. As it is expected, SORT and RANDOM WS strategies resulted noticeably higher blocking probability in case of any offered load value than the sophisticated methods. On the other hand, it is very interesting that EXHAUST(4) produces poor quality solution. This is because EXHAUST(4) takes only the residual fiber information into account during the WS process resulting

roundabout paths on the selected wavelength, and later this long paths cause blocking of other demands. The other four types of EXHAUST strategy result about the same blocking probability curves, using any of them results good network performance.

We can introduce the so-called *wavelength selection method gain* denoted by γ , that is a proportion number, which can measure the quality of the wavelength selection methods. It can show the steady state difference between the blocking probabilities resulting two different WS strategies. It can be computed in the following way: $\gamma_{1,2} = 1 - \frac{\mathcal{P}_b[WS_method(1)]}{\mathcal{P}_b[WS_method(2)]}$ (in percent), where in both methods the routing algorithm use the same weight function. In the same way we introduce a proportion number to measure the quality of a weight function, denoted by δ . The computation is very similar to the previous case, $\delta_{1,2} = 1 - \frac{\mathcal{P}_b[weight_func(type1)]}{\mathcal{P}_b[weight_func(type2)]}$ (in percent) and we assume that the WS methods are the same in case of both weight functions. $\delta_{1,2}$ gives us information how much is the steady state difference between the blocking probabilities using different weight functions. Tables 4.2 and 4.3 contain information about the wavelength selection method gain (γ) in case of 55% and 75% offered load using *Net_{AT&T}*.

Table 4.2: Gain of WS strategies - 55% offered traffic

Load	γ_{SORT_SORT}	γ_{RANDOM_SORT}	$\gamma_{EXHAUST(0)_SORT}$	$\gamma_{EXHAUST(1)_SORT}$
0.55	-	-2.64%	27.829%	27.255%
Load	$\gamma_{EXHAUST(2)_SORT}$	$\gamma_{EXHAUST(3)_SORT}$	$\gamma_{EXHAUST(4)_SORT}$	
0.55	27.37%	24.503%	3.82%	

Table 4.3: Gain of WS strategies - 75% offered traffic

Load	γ_{SORT_SORT}	γ_{RANDOM_SORT}	$\gamma_{EXHAUST(0)_SORT}$	$\gamma_{EXHAUST(1)_SORT}$
0.75	-	-3.47%	9.595%	7.86%
Load	$\gamma_{EXHAUST(2)_SORT}$	$\gamma_{EXHAUST(3)_SORT}$	$\gamma_{EXHAUST(4)_SORT}$	
0.75	8.91%	8.717%	1.93%	

On the basis of the γ values we can see that in case of 55% offered traffic, EXHAUST(0) strategy produced the best result (it produces 27.829% less blocking probability than the SORT method). EXHAUST(1) and EXHAUST(2) also perform very impressive results while EXHAUST(3) produces a little bit worse values.

In case of larger offered traffic we can realize the same tendency between performance of the WS strategies than in the previous case. Furthermore, EXHAUST(0)

produces the best result, but the difference between this and the other three (1,2,3) EXHAUST type methods becomes a little bit larger. On the other hand, we can realize that the γ values are smaller than in the previous case, which is caused by the increased traffic to be carried by the network.

The effectiveness of the operation in cases of different weight functions is also very important measure of the WS&WR method. Therefore it is tested how the blocking probability varies with different kinds of weight function and our aim was also to determine which is the most recommended weight function, which can perform a good network operation for both moderate and heavy load. We used the same action_lists as in the previous tests, and for wavelength routing strategy we selected EXHAUST(0) since it guarantees the best WS solution. Figure 4.2 shows the resulted blocking probability functions.

If we consider these three figures together, we can see how the effectiveness of the different weight functions depend on the size of the network and number of the traffic demands. First of all, it can be seen that weight type 6 performs significantly worse blocking probability than the other methods. (This is because weight type 6 does not take any path length information into account.) We do not expect that weight type 1 produces good results, because this does not take any load value (number of free fibers on the adequate wavelength) into account. In case of Net_{PI} there is no significant difference between the weight functions except of 6, especially when the offered traffic is near to 80-90%. Between 0.5 and 0.65 offered traffic, weight type 3 performs the best blocking probability, while in case of greater traffic intensity weight type 2 is the best, but 7 and 8 also provide a good overall blocking probability. Weight types 4 and 5 do not work optimally in case of a less loaded network, but its performance is increasing with the traffic intensity as can be realized in the 0.75 - 0.95 offered traffic interval. We have to deal with weight type 4 because it produces a not very favorable curve. Considering the formula of weight types 3 and 4 we can realize that the difference between them is almost negligible, but the curves resulted from these formulas significantly differ. In case of $Net_{AT\&T}$ we meet similar situation than in the previous network. Weight type 2 is the best, but weight types 3,5,7 and 8 also perform reasonable and stable blocking in case of any offered traffic value. Using Net_{56} almost all curves can be distinguished inside the whole examined offered traffic area, but there is no absolutely optimal weight type. When the offered traffic is between 0.5 and 0.65 weight type 5, when 0.7 weight type 3, in case of 0.75 weight type 5 again and finally between 0.8 and 0.95 weight type 2 produces the best blocking probability values. Similarly, weight types 7 and 8 also perform quite good results. Summarizing, weight types 2,3,5,7 and 8 can guarantee a reasonable blocking probability in case of any network size and traffic intensity. A further, fruitful research direction could be how to combine

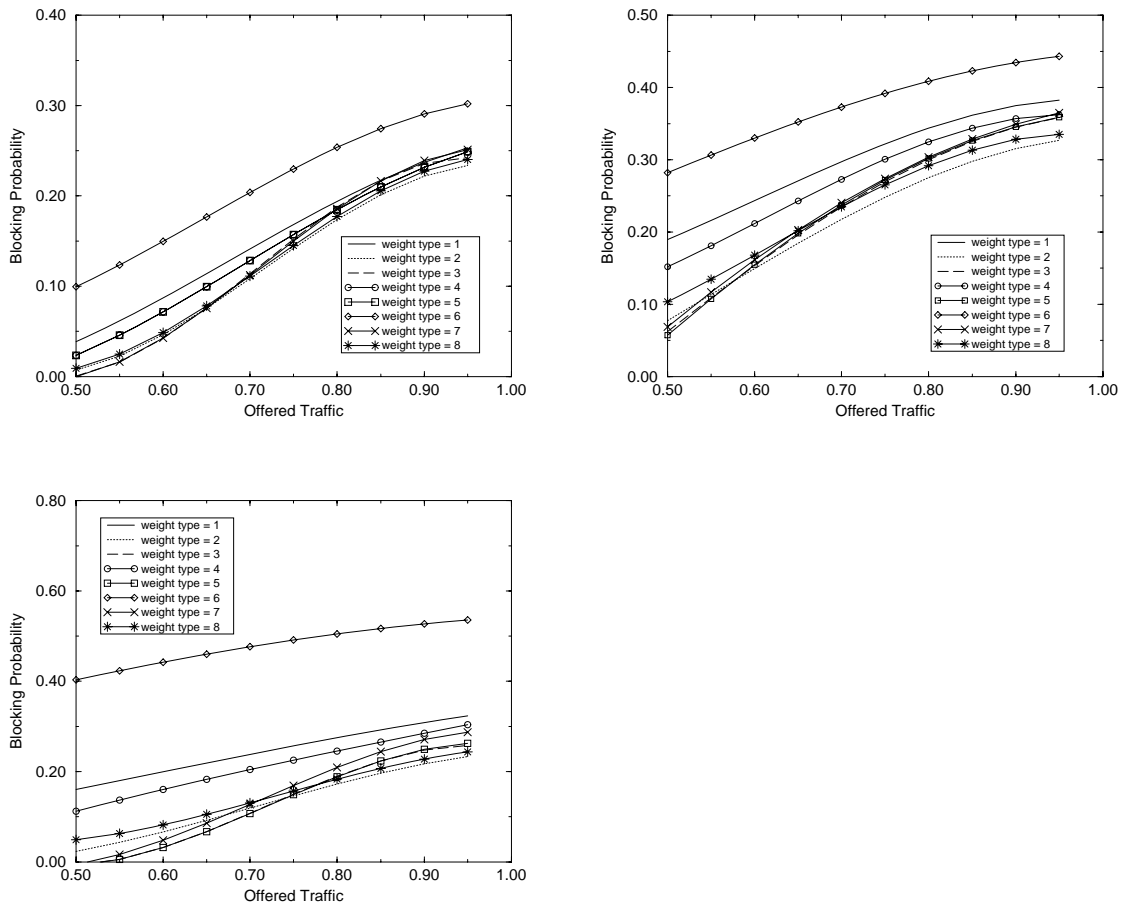


Figure 4.2: Blocking probability using different weight functions in Net_{PI} , $Net_{AT\&T}$ and Net_{56}

these weight functions (using different function in case of different traffic intensity and network state) to find the best overall solution.

Tables 4.4 and 4.5 contain information about the wavelength selection method gain (δ), in case of 55% and 75% offered load using $Net_{AT\&T}$.

Table 4.4: Gain of weight functions - 55% offered traffic

Load	δ_{1_1}	δ_{2_1}	δ_{3_1}	δ_{4_1}
0.55	-	43.257%	44.405%	13.722%
Load	δ_{5_1}	δ_{6_1}	δ_{7_1}	δ_{8_1}
0.55	45.848%	39.19%	42.727%	34.953%

Table 4.5: Gain of weight functions - 75% offered traffic

Load	δ_{1_1}	δ_{2_1}	δ_{3_1}	δ_{4_1}
0.75	-	21.531%	15.614%	8.957%
Load	δ_{5_1}	δ_{6_1}	δ_{7_1}	δ_{8_1}
0.75	13.335%	-18.453%	14.994%	17.373%

We have two notes on the above tables. At first, we can realize how δ decreases as the offered traffic intensity increases. On the other hand, it seems that in case of 55% offered traffic weight type 5 is the best, but in case of 75% weight type 2 is the winner. It is also important that if we use sophisticated methods, we can achieve 40-50% less blocking probability than using simple shortest path and δ remains about 10% even in a heavy loaded network.

Beside the blocking probability, it is also important how the network load is changing with the increasing of offered traffic. That is why we calculate the network load using $Net_{AT\&T}$ in case of different WS processes combined with weight type 3, and different weight types with EXHAUST(0) WS method to examine the performance of our algorithm from this viewpoint, too (Figure 4.3).

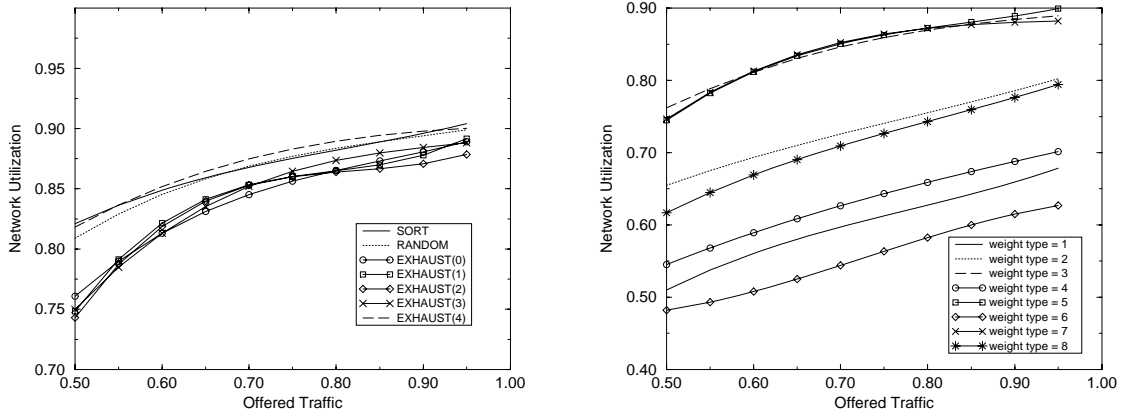


Figure 4.3: Network utilization using different WS strategies and weight functions in $Net_{AT\&T}$

The curves on the two figures seem to be very different, however we can find the connection between them. On the first figure we can find the two groups of curves as in Figure 4.1. The sophisticated methods result less network load

when the offered traffic are relatively small, but the simpler WS strategies (SORT, RANDOM, EXHAUST(4)) result higher network load even in case of smaller traffic intensity. With the increase of the offered traffic, the curves get closer to each other, finally the utilization begins to decrease caused by an overloaded network situation when the complicated WS strategies cannot already produce an optimal load balancing in the network, neither. On the other figure weight types 1, 4 and 6 result low network utilization because of their high blocking probability. Weight types 3, 5 and 7 produce very similar curves as on the other figure while weight types 2 and 8 result a "middle" level utilization. Since the largest value of the utilization using weight type 2 is about 80% it is sure that this type of function provides reasonable blocking probability even in case of larger traffic intensity. Weight types 3, 5 and 7 result utilization close to 90%, consequently they cannot handle significant greater traffic demands to be established.

4.4.3 Issues on Fairness Criteria

In this Section we investigate how the overall blocking probability P_b is influenced if the fairness between the users is considered. That is why we compare the value of P_b and the variance of p_i values (the variance of user level blocking probability) when the fairness process is used and when it is not. We use *Net_{AT&T}*, EXHAUST(0) WS method and weight type 3. We perform this investigation in case of different demand arrival intensity, from about 50% of the demands are active to 90% of the demands are active. Variance of p_i values describes the deviation of p_i values from P_b in percent. In this tests 150000 demand requests are generated to obtain the required number of individual user requests to compute p_i values. The results are summarized in Table 4.6 and 4.7.

Table 4.6: P_b and variance of p_i values if fairness is not considered

Network load [%]	P_b [%]	Variance of p_i values[%]
50	4.51	4.45
60	11.82	5.26
70	13.54	6.39
80	19.63	7.72
90	26.76	8.36

I have two main notes on the above Tables. If we see Table 4.6, we may say that beside a relative advantageous P_b value an individual p_i value can be larger than the overall blocking probability P_b . On the basis of Table 4.6 we may say that the variance of p_i values increase with the value of P_b if the fairness is not considered. If the fairness between the users is provided we can see that the variance values

Table 4.7: P_b and variance of p_i values if fairness is considered

Network load[%]	P_b [%]	Variance of p_i values[%]
50	4.85	2.24
60	12.76	2.83
70	15.17	3.06
80	22.07	3.18
90	29.73	3.79

are significant smaller and do not depend on P_b so much. The price of ensuring the fairness is the slight increasing of the overall blocking probability.

4.5 Application of the WS&WR algorithm in WDM network configuration

In this section I will propose a possible way to apply the WS&WR algorithm in WDM network configuration/dimensioning tasks. The task to be solved is to calculate the required network resources (wavelength and fiber number) on the way, which results low blocking probability in case of a given demand arrival process.

The concrete planning task that we solve is the following:

- The network topology structure is given, but neither the number of fibers contained by a link l (denoted by $F_{num}(l)$, $l \in \mathcal{L}$) nor the number of wavelengths in the network (denoted by W_{num}) are not known.
- We are given a traffic matrix, which contains the intensity of connection establishment requests and the expected average holding time of the demands between any node pair. It means that we assume that the demand arrival process is known.
- The goal is to find such $F_{num}(l)$ values for all link and W_{num} values for the whole network that provides that the long-term blocking probability will be less than a pre-defined value (denoted by P_{goal}) in case of the above demand incoming process. By other words the ultimate goal is to plan a network, which fits to the dynamic traffic arrival process as best as possible.

I proposed a method, which basically built on the WS&WR algorithm; the method works according to the following process:

Initial Step The $F_{num}(l)$ values and W_{num} value will be set to 1.

Step 1. Using the current network resources (number of fibers and wavelengths) with the help of the incoming demand process a working network is simulated.

Then using the WS&WR algorithm we try to allocate the incoming demands in the network, and during this establishment process a current blocking probability (P_{curr}) is measured. When the network obtains its steady state operation mode we got the long-term value of P_{curr} . Then the difference between P_{goal} and P_{curr} is calculated. If $P_{goal} < P_{curr}$, it means that the current network resources are not enough to provide the pre-defined network performance, consequently some kind of network resource extensions are required. In the another case, when $P_{goal} \geq P_{curr}$, the network has enough resources, the optimization is finished at *Final Step*.

Step 2. – Network Extension The network resources can be extended in two ways, namely:

- Increasing the number of fibers: Because the number of fibers can be different link by link it is possible to apply this modification only on those links that cause the great part of the unsuccessful demand allocation attempts. Therefore the algorithm stores how many times a link was bottleneck during the establishment process and these links will be extended in different degree. Of course it is possible to extend all the links in the same way.
- Increasing the number of wavelengths: According to the network model this modification affects the whole network; the number of available wavelengths will be increased all the links with the same value.

Based on the aboves, F_{num} or W_{num} , as well as both of them can be increased in the following way:

- $F_{num}(l) = F_{num}(l) + n(l)$ for all link $l \in \mathcal{L}$
- $W_{num} = W_{num} + m$
- $F_{num}(l) = F_{num}(l) + n(l)$ for all link $l \in \mathcal{L}$, and $W_{num} = W_{num} + m$

Here $n(l)$ and m are integer numbers, which can be set by the user, or can be set adaptively during the optimization according to the difference between P_{goal} and P_{curr} (the value of $n(l)$ and m are decreasing with this difference), and the bottleneck parts of the network. After the network extension the optimization is continue at Step 1.

Final Step The optimization arrives here when the network resources are enough to provide P_{goal} in the long-term. If in any phase during the optimization $n(l)$ or m was greater than 1, it is possible that the final network is overdimensioned (which means that it contains more resource than the required for the given P_{goal}). Therefore in this final step the algorithm attempts to set all $F_{num}(l)$ and W_{num} values as fine as possible. The resources will be decreased one by one and – as in *Step 1*.

a working network will be simulated, and the steady state value of P_{curr} is checked continuously. While P_{curr} is less than P_{goal} the resource decreasing is continuing, which results that the network can be dimensioned in a very precise way for the pre-defined blocking probability in case of a known arrival process of the incoming demands. The optimization is finished with the last network configuration, where P_{curr} is valid.

The above method can also be used to solve the classical static dimensioning task. The problem is here the following: We have a final set of demands and we find such configuration of network resources and such path system for the demands (allocation of a path on a wavelength), which provides that *all* of the demands are routed.

Solving this task, P_{goal} is set to 0, which means that we find such a solution where all the demands installed into the network, what is a possible static configuration of the network. Of course it is absolutely not sure that the first static configuration found is the best one, therefore the algorithm repeats the above process to find some other valid network configurations. Then the algorithm selects between the most resource-saved configurations that one, which results the best network utilization (which contains the most free resource in case of the same $F_{num}(l)$ and W_{num}) values.

I would like to illustrate the effectiveness of the application of my proposed WS&WR method in static network configuration. Therefore I compared my method with a known *Dimensioning/Configurator Algorithm (DCA)* presented in [35].

In the first test I examined how many network resources needed to route the same number of demands. In this context the resource means the sum of the number of links used in the network in case of the same wavelength number. By other words we find such a network configuration where $\sum_{l \in \mathcal{L}} F_{num}(l)$ is minimal in case of the same W_{num} value. In case of all the tests the needed resources in the results of the DCA was 100% and the results of WS&WR was compared to this value (in percent). Eight sizes of networks are used, in case of each size 20 different random generated topologies are tested. The results are summarized in Table 4.8.

The WS&WR algorithm needs averagely 8.4% less resources to route all the demands in the network. It is very favourable result, but I have to note that the value resource saving very depends on the topology of the network.

In the second test I examined that in case of same resources (e.g. same value of $F_{num}(l)$ and W_{num}) which algorithm is able to offer such path configuration of the demands that results the larger value of the residual capacity in the network. All the network was configured with the DCA method and the resources were increased so that the residual resources (e.g. wavelengths) is about 4–7%. The results (residual resource values obtained by the DCA and WS&WR) can be seen

in Table 4.9.

Table 4.8: Effectiveness of the Ws&WR method in static configuration

#Nodes	DCA	WS&WR
50	100	93.2
70	100	90.8
100	100	91.3
120	100	95.8
150	100	92.1
200	100	89.5
300	100	90.5
500	100	89.4

Table 4.9: Effectiveness of the Ws&WR method in static configuration

#Nodes	DCA	WS&WR
50	4.5	9.4
70	4.3	9.9
100	5.6	9.1
120	5.7	10.8
150	5.1	9.2
200	6.7	11.5
300	5.6	10.9
500	4.8	10.4

The above results prove that applying the WS&WR method we can find such a path system, which provides averagely 4.86% more free resources in the network, which is a measurable improvement, especially in case of large size networks.

4.6 Conclusions

This chapter studied the dynamic Wavelength Selection and Wavelength Routing (WS&WR) problem in multifiber optical networks. A suitable model has been developed for circuit-switched type WDM networks, where the main task was to find an appropriate wavelength (wavelength selection task – WS) and an optimal path (routing task – WR) for the incoming demands. Sophisticated strategies for the WS problem have been proposed, which take the current network state into account, while in the routing problem different kinds of link state dependent weight functions have been compared. We have analyzed these methods in different real

networks and prepared a detailed performance analysis of them in term of steady-state blocking probability. The tests of the WS&WR algorithm indicate that the network state dependent WS strategies, as well as the link state dependent weight functions result better network performance (blocking probability) than the simple sort and random WS methods and the classical hop count and/or load based weight functions. For example, EXHAUST(0) WS strategy performs about 10-27% less blocking probability than the often used sorting and random methods. Similarly, the proposed best weight function (2) results 20-40% less blocking probability than the fixed or alternate routing. Summarizing, by applying the proposed algorithms, the performance of optical WDM backbone networks improves significantly.

Application of the Results

The algorithms presented in Thesis 1 are used in the UMTS Network Planning and Analysis Tool of Ericsson Telecommunication Ltd. The results of the UMTS core network planning research (Thesis 2) was sponsored by a Product Unit of Ericsson Hungary, and they will be used in a Network Configurator Tool. The OSPF related research was carried out in the frame of a cooperation between Ericsson Research Hungary and Warsaw Technical University. The dynamic path establishment algorithm in WDM network related work was a result of a cooperation between the Ericsson and the High Speed Networks Laboratory of Budapest University of Technology and Economics and it will be used as a configurator method in WDM Network Planning Tool.

Bibliography

- [1] T. Ojanperä, R. Prasad, *Wideband CDMA for Third Generation Mobile Communication*, 1998, Artech House Publishers.
- [2] Z. Drezner (editor), *Facility Location*, Springer Series in Operations Research, 1995
- [3] R. Metwani, P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995
- [4] P. M. Camerini, G. Galbiati, F. Maffioli, *Complexity of Spanning Tree Problems*, European Journal of Operational Research, 1980.
- [5] P. M. Camerini, G. Galbiati, F. Maffioli, *On the Complexity of Finding Multi-constrained Spanning Trees*, Discrete Applied Mathematics vol.5, 1983
- [6] S. C. Narula, C. A. Ho, *Degree-constrained minimum spanning tree*, Comput. Ops. Res. 7. 1980.
- [7] B. Boldon, N. Deo, N. Kumar, *Minimum-weight Degree-Constrained Spanning Tree Problem: Heuristic and Implementation on an SIMD parallel machine*, Technical Report, 1995, Dep. of Computer Science, University of Central Florida, Orlando
- [8] S. Dravida, H. Jiang, M. Kodialam, B. Samadi, Y. Wang, *Narrowband and Broadband Infrastructure Design for Wireless Networks* IEEE Communications Magazine, May, 1998.
- [9] P. Kallenberg, *Optimization of the Fixed Part GSM Networks Using Simulated Annealing*, Proc. Networks 98, 8th International Telecommunication Network Planning Symposium, Sorrento, Italy, October 1998.
- [10] B. Hajek, *A Tutorial Survey of Theory and Applications of Simulated Annealing*, Proc. 24th Conference on Decision and Control, Ft. Lauderdale, FL., December 1985.

- [11] B. Hajek, *Cooling Schedules for Optimal Annealing*, Mathematics of Operation Research, Vol 13, No.2, May 1988
- [12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison Wiley, 1989
- [13] P. Gajowniczek, M. Pióro, A. Arvidsson, *VP reconfiguration through Simulated Allocation*, NTS 13, Thirteenth Nordic Teletraffic Seminar, August 1996.
- [14] B. Fortz, M. Labbé, F. Maffioli, *Methods for Designing Reliable Networks with Bounded Meshes*, 15th ITC (International Teletraffic Congress), 1997.
- [15] M. Zethzon, T. Cinkler, I. Andersson, *Greedy Algorithms for Topological Design* Proc. Networks 98, 8th International Telecommunication Network Planning Symposium, Sorrento, Italy, October 1998.
- [16] K. Holmberg, J. HellStarnd, *Solving the Uncapacitated Network Design Problem by a Lagrangean Heuristic and Barnch and Bound* Operations Research 46: 247-259, 1998.
- [17] *OSPF Version 2*. RFC2328.txt, www.ietf.org/rfc/rfc2328.txt, 1998
- [18] B. Fortz, M. Thorup, *Internet Traffic Engineering by Optimizing OSPF Weights*, Proc. INFOCOM 2000, Tel-Aviv, 2000
- [19] W. B. Ameur, E. Gourdin, B. Liau *Internet Routing and Topology Problems*, Proc. DRCN 2000, Design of Reliable Communication Networks, Munich, 2000
- [20] W. B. Ameur, E. Gourdin, B. Liau *Dimensioning of Internet Networks*, Proc. DRCN 2000, Design of Reliable Communication Networks, Munich, 2000
- [21] A. Faragó, B. Szviatovszki, Á Szentesi, *Allocation of Administrative Weights in PNNI*, Proc. Networks 98, 8th International Telecommunication Network Planning Symposium, Sorrento, Italy, October 1998.
- [22] G. Kotelly, *Worldwide fiber-optic markets to expand unabated*, Lightwave, Vol. 13, No. 13, pp. 6-8, December 1996
- [23] I. Chlamtac, A. Ganz, G. Karmi, *Lightpath communications: an approach to high bandwidth optical WAN's*, IEEE Transactions on Communications, Vol. 40, no. 7, 1992
- [24] S. Xu, L. Li, S. Wang, *Dynamic Routing and Assignment of Wavelength Algorithms in Multifiber Wavelength Division Multiplexing Networks*, IEEE Journal on Selected Areas in Communications, Vol. 18 No. 10, October 2000

- [25] X. Zang, C. Qiao, *Wavelength Assignment for Dynamic Traffic in Multi-fiber WDM Networks*, ICCCN'98, Vol. 18 No. 2, Lafayette, LA, 1998
- [26] E. Karasan, E. Ayanoglu, *Effects of wavelength routing and selection algorithms on wavelength conversion gain in WDM optical networks*, Proc. IEEE GLOBECOM, London, England, November 1996
- [27] A. Mokhtar, M. Azizoğlu, *Adaptive wavelength routing in all-optical networks*, submitted to IEEE/ACM Transactions on Networking
- [28] S. Subramaniam, R. Barry, *Wavelength assignment in fixed routing WDM networks*, Proc. IEEE ICC, Montreal, Canada, November 1997
- [29] A. Birman *Computing approximate blocking probabilities for a class of all-optical networks*, Proc. IEEE INFOCOM, April 1995
- [30] H. Harai, M. Murata, H. Miyahara, *Performance Of alternate routing methods in all-optical switching networks*, Proc. INFOCOM, April 1997
- [31] L. Li, A. K. Somani, *Dynamic Wavelength Routing Using Congestion and Neighborhood Information*, IEEE/ACM Transactions on Networking, Vol. 7, No. 5, October 1999
- [32] Nilesh M. Bhide, Krishna M. Sivalingnam, *Routing mechanisms employing adaptive weight functions for shortest path routing in optical WDM networks*, Photonic Network Communications, Vol. 3. No. 3. 2001
- [33] E. Karasan, S. Banerjee, *Performance of WDM Transport Networks*, IEEE Journal on Selected Areas in Communications, Vol. 16, p. 764-779, September 1998
- [34] E. Karasan, E. Ayanoglu, *Performance comparison of reconfigurable wavelength selective and wavelength interchanging cross-connect in WDM transport networks*, Proc. 3rd IEEE/COMSOC Workshop on WDM Network Management and Control, Montreal, Canada, November 1997
- [35] G. Conte, M. Listanti, R. Sabella, M. Settembre *Strategy for protection and restoration of optical paths in WDM backbone networks for next generation Internet infrastructures*, accepted for publication on Journal of Lighthwave Technology, 2002

Publications

Journal papers

- [J1] J. Harmatos *Számlázás ATM rendszerben*, Magyar Távközlés, 9. évfolyam, 5. szám, 1998 Május, pp. 21–26
- [J2] J. Harmatos *A hálózatok fizikai rétegének méretezése*, Magyar Távközlés, 9. évfolyam, 11. szám, 1998 November, pp. 12–15
- [J3] G. Bóné, J. Harmatos *Algorithms in Network Planning and Management*, Hungarian Telecommunications Periodical, Selected Papers of 1999, 1999, pp. 33–38
- [J4] G. István, J. Harmatos, *Az UMTS hozzáférési hálózatának tervezése*, Magyar Távközlés, 2000 Augusztus
- [J5] M. Pióro, Á. Szentesi, J. Harmatos, A. Jüttner, P. Gajowniczek, S. Kozdrowski: *On OSPF Related Network Optimisation Problems*, Performance Evaluation, Vol 48, Nr 1–4, May 2002, pp. 201–223
- [J6] A. Szlovensák, I. Gódor, J. Harmatos, T. Cinkler: *Planning Reliable UMTS Terrestrial Access Networks*, IEEE Communications Magazine, January 2002, pp. 66–72
- [J7] J. Harmatos, P. Laborczi: *Dynamic Routing and Wavelength Assignment in Survivable WDM Networks*, Photonic Network Communications, Vol. 4, Nr. 3–4, July–December 2002, pp. 357–375

Conference and workshop papers

- [C1] J. Harmatos, A. Jüttner, Á. Szentesi: *Cost-based UMTS Transport Network Topology Optimization*, International Conference on Computer Communication, ICC'99, Tokyo, Japan, September 1999, pp. 00111–1–8
- [C2] M Piòro, Á Szentesi, J. Harmatos, A. Jüttner, S. Kozdrowski: *On OSPF Related Networks Optimization Problems*, 8th IFIP Workshop on Performance

Modelling and Evaluation of ATM & IP Networks, Ilkley, UK, July 17–19, 2000, pp. 70/1–70/14

- [C3] A. Jüttner, J. Harmatos, Á. Szentesi, M. Piòro: *On Solvability of an OSPF Routing Problem*, 15th Nordic Teletraffic Seminar, Lund, Sweden, August 22–24, 2000, pp. 1–9
- [C4] J. Harmatos, Á. Szentesi, I. Gódor: *Planning of Tree-Topology UMTS Terrestrial Access Networks*, 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2000, London, UK, September 18–21, 2000, pp. 353–357
- [C5] P. Gajowniczek, M. Piòro, Á. Szentesi, J. Harmatos: *Solving an OSPF Routing Problem with Simulated Allocation*, First Polish–German Teletraffic Symposium, PGTS 2000, Dresden, Germany, September 24–26, 2000, pp. 177–184
- [C6] A. Jüttner, J. Harmatos, D. Orincsay, B. Szviatovszki, Á. Szentesi: *On-demand Optimization of Label Switched Paths in MPLS Networks*, In proc. of the 9th International Conference on Computer Communications and Networks, IEEE ICCCN 2000, Las Vegas, USA, October 16–18, 2000, pp. 107–113
- [C7] D. Orincsay, J. Harmatos: *On-demand Optimization of Protected LSP Tunnels in MPLS Networks*, 9th IFIP Conference on Performance Modelling and Evaluation of High Speed Networks, Budapest, Hungary, June 27–29, 2001, pp. 357–368
- [C8] A. Szlovenscák, I. Gódor, T. Cinkler, J. Harmatos: *Planning Reliable UMTS Terrestrial Access Networks*, 3th International Workshop on Design of Reliable Communication Networks, DRCN 2001, Budapest, Hungary, October, 2001, pp. 84–90
- [C9] M. Piòro, A. Jüttner, J. Harmatos, Á. Szentesi, P. Gajowniczek, A. Myslek: *Topological Design of Telecommunication Networks*, 17th International Teletraffic Congress, ITC 17, Salvador da Bahia, Brazil, December 2–7, 2001, pp. 629–642
- [C10] M. Piòro, A. Jüttner, J. Harmatos, Á. Szentesi, A. Myslek: *Topological Design of MPLS Networks*, GLOBECOM 2001, San Antonio, USA, 25–29 November, 2001, pp. 12–16
- [C11] J. Harmatos: *A Heuristic Algorithm for Solving the Static Weight Optimization Problem in OSPF Networks*, GLOBECOM 2001, San Antonio, USA, 25–29 November, 2001, pp. 1605–1609

- [C12] G. Salamon, S. Györi, J. Harmatos, T. Cinkler: *Dimensioning WDM-based Multi-layer Transport Networks with Grooming by Genetic Algorithm*, 7th European Conference on Networks & Optical Communications, NOC 2002, Darmstadt, Germany, 18-20 June, 2002
- [C13] J. Harmatos: *Planning of UMTS Core Networks*, 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2002, Lisboa, Portugal, September 15–18, 2002