Available online at www.sciencedirect.com
SCIENCE DIRECT
Computer Communications 28 (2005) 765–772
computer communications
www.elsevier.com/locate/comcom

Inverse shortest path algorithms in protected UMTS access networks

István Gódor*, János Harmatos, Alpár Jüttner

Ericsson Research, Traffic Analysis and Network Performance Laboratory, P.O. Box 107, H-1300 Budapest, Hungary

Received 26 November 2003; revised 28 October 2004; accepted 28 October 2004
Available online 23 November 2004

Abstract

In this paper, the application questions of Open Shortest Path First (OSPF) routing protocols in the all-IP based protected Universal Mobile Telecommunications System (UMTS) access networks is investigated. The basic problem here is how the OSPF administrative weights should be adjusted in an adequate way, resulting near-optimal overall network performance both in nominal network operation and in case of single link failures. Currently, there are known algorithms which are able to solve the topology planning and dimensioning related problems of UMTS access networks, but they do not consider the special properties of the OSPF protocol. We formulate the problem as an Integer Linear Programming (ILP) task. Furthermore, we propose an advanced heuristic algorithm and compare its results with the ILP solution and other existing algorithms.

Keywords: OSPF; UMTS; Traffic engineering

1. Introduction

In this paper, we deal with configuration of Open Shortest Path First (OSPF)-based fault-tolerant UMTS (Universal Mobile Telecommunications System) Terrestrial Radio Access Networks (UTRAN) [1,2]. The OSPF-based routing [3,4] in the initial releases of UTRAN is a trivial task because the topology determines the paths by itself. However, the complex topology of fault-tolerant networks makes the OSPF configuration complicated. First the UTRAN and OSPF will be introduced, then the fault-tolerant related issues will be presented.

UTRAN networks contain two main types of equipment, namely the Radio Base Station (RBS), and the Radio Network Controller (RNC). The task of the RBS is to handle both the user and control radio channels of the particular cell, as well as to concentrate the traffic of lower level RBSs and forward it towards its dedicated RNC. The role of the RNC is managing the RBSs that are connected to it, and concentrate the traffic flows of connections and forward them to the upper level core network. The topology of the UMTS access network is a set of trees, in which the RBSs form the tree, and the RNC is the root node. The different trees are connected to each other via the RNCs typically using a mesh-like topology based on another transmission technology. The network is modeled as a set of trees and represented by a graph, where the nodes are the vertices and the links are the edges.

Because of the original tree-like topology of UMTS terrestrial access networks, these networks are very sensitive to the failures and in case of larger networks, even a single failure can cause a loss of high amount of data. One solution is to use fault-tolerant topologies, such as ring or mesh like topologies, however, their cost can be significantly greater than the simple tree topology. A possible alternative solution is to keep the basic tree topology and expand it with some additional links at those parts of the network where faults are critical. The main advantage of the network extension is that an acceptable equilibrium can be found between the increases of the cost and fault tolerance capability of the network.

Solving this kind of network extension problem is not an obvious task, because the state space of the problem is large and the goal is twofold, we want to increase the fault-tolerance with minimal investment. We analyze

* Corresponding author. Tel.: +36 1 437 7237; fax: +36 1 437 7767.
E-mail addresses: istvan.godor@ericsson.com (I. Gódor), janos.harmatos@ericsson.com (J. Harmatos), alpar.juttner@ericsson.com (A. Jüttner).

0140-3664/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2004.10.011

the properties of this kind of network extension solution [5] and propose a planning algorithm to this problem. It is shown that our algorithm gives high quality solutions.

In the first phase the transport technology of the UMTS terrestrial access networks will be Asynchronous Transfer Mode (ATM) based, but the ultimate goal is to apply IP technology here as well. Because the most widespread routing protocol of IP networks is OSPF protocol, it seems to be used in UTRANs too. The OSPF uses the shortest path for routing packets according to the given weights of the links and it may apply the so-called Equal-Cost Multipath (ECMP) principle in cases of multiple shortest paths. The OSPF routing procedure works essentially as follows: A positive integer number is assigned to each link (called weight), and these values are sent to the network nodes using the OSPF link-state flooding mechanism. Every intermediate node along a path determines the next hop of the packet towards the destination node using local routing table. Building up the routing tables of the nodes, the shortest paths are calculated on the basis of the link weights.

Our model used in [5] is able to handle only the explicitly defined paths (this is the case in ATM based access network), and cannot consider the special properties of OSPF routing. We keep this algorithm to determine the position and capacity of the extra links required to obtain the defined protection level, and to determine the routes of protection paths along with the required capacity increment (if any). In this case, the problem is that the backup paths are determined explicitly, therefore, we need a method, which computes an OSPF weight system reproducing the original paths.

The rest of the paper is organized as follows. First the network model and the task specification is presented. Then two methods, a heuristic-based and a LP-based method are outlined. Then the heuristic algorithm is compared with the LP solution and an existing reference method. We conclude the paper with some computational results and summarize our experiences.

2. Network model and task specification

The goal of this section is to present the used network model, as well as the clarification of the exact optimization problem to be solved and the considered conditions.

The network is modeled as an undirected graph $G(V, E)$, where vertex set V represents the node of the network, while edge set E corresponds to the links. There is a distinct node $r \in V$ that corresponds to the RNC.

The network topology is a spanning tree completed with some additional links to provide the pre-defined level of network availability. So, we distinguish the two types of links, namely the links belonging to the spanning-tree are called *default links*, while the additional links are called *backup links*. The set of default links is denoted by E_d and the set of backup links is denoted by E_b.

The tree E_d determines the *level* $l(v)$ of each node v, i.e. the distance of v and r on this tree.

We require that exactly one default link and at most one backup link can be originated from each node. Moreover, the backup links can connect only two nodes on the same level or on adjoining upper levels (for more information about the technological background see [5]).

We are also given a set of the protected failure scenarios as a set of edge sets $E^i \subseteq E$, $i \in \{0,1,2,\ldots,k\}$ expressing the available edges in the ith failure scenario. E^0 corresponds to the case when there is no link failure, i.e. $E^0 = E_d$.

For each node v and for each failure scenario i we are given a path $p_v^i \subseteq E^i$. This path is used to carry data from v to r in case of the ith failure scenario. The path p_v^0 is called *default path* the others are the *backup paths*. If the node is not protected against a failure then $p_v^i = \varnothing$. We also require to use the working path whenever it is possible, i.e. if $p_v^0 \subseteq E^i$ then $p_v^i = p_v^0$.

A positive integer number $w(e)$ is assigned for each link $e \in E$ as the OSPF weight. We can similarly define the weight of a path: $W(p_v^i) = \sum_{e \in p_v^i} w(e)$.

In this paper, only *one* link failure is considered at a time (which is relevant to the generally accepted network modeling). For the sake of simple configuration, we require that each backup path contains only *one* backup link and some default links. This ensures that the backup paths will not be too long.

The optimization proceeds from a network with known topology, link capacities and paths. If reproducing the predefined path system is possible, then the output is an adequate OSPF weight for each link. If there exists no such a weight system, then we propose where and how to modify some backup paths to achieve an OSPF conform path system. As a consequence of OSPF routing, all used paths in a given state (normal operation or any of the failure-cases) must form a tree.

3. Proposed heuristic algorithm

Our proposed solution is to combine the strategy of path 'restoration' with the strategy of overload decrement. Thus if the predefined path system cannot be reproduced for some reason, then it can still minimize the overload of the links. The framework of the proposed *OSPF Weight Setting (OWS) algorithm* is illustrated in Fig. 1.

Fig. 1 shows that the OWS algorithm iteratively uses five procedures (for details see the following subsections) in order to solve the inverse shortest path problem in the following way:

1. Initialization:
   (a) Initialize the weights of the links (by procedure IWS).
   (b) Search for infeasible situations caused by circles of backup links. If it is possible to decompose

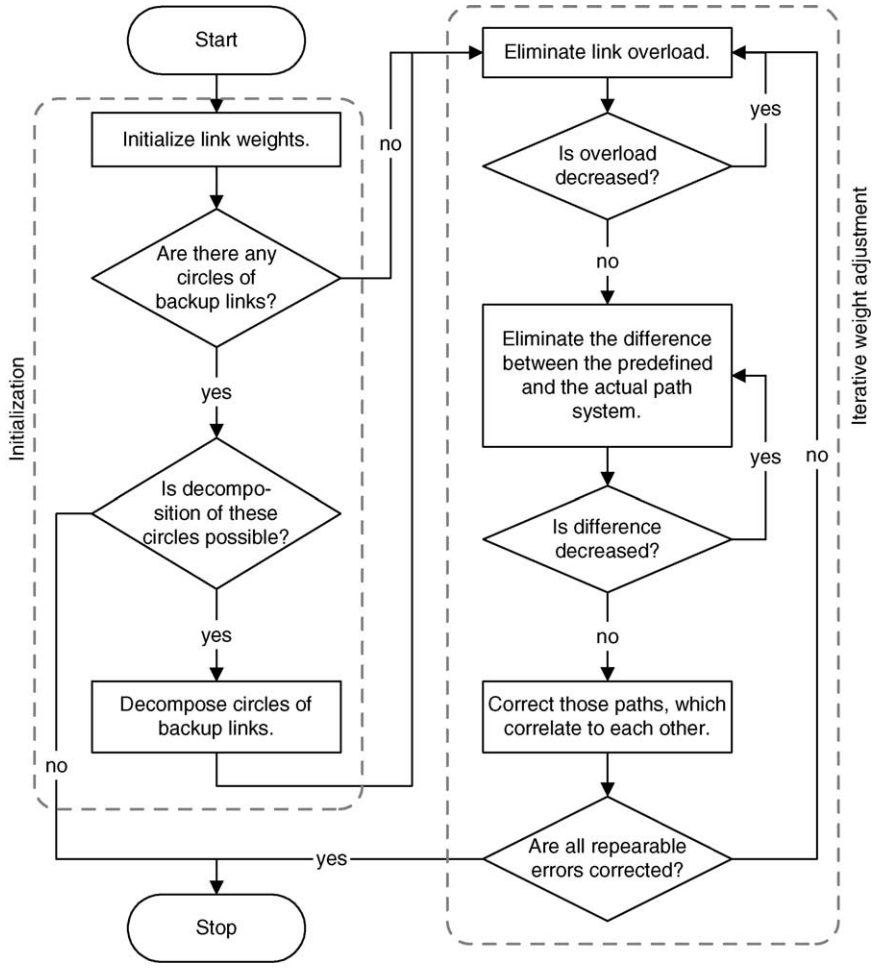

Fig. 1. The framework of the OWS algorithm.

the circles, then decompose them (by procedure DEC_1 or DEC_2), else STOP.

2. Iterative weight adjustment:
   (a) Eliminate the link overload in the network by modifying the weight of the links (by procedure BWS), and repeat this step until the overload decreases.
   (b) Eliminate the difference between the predefined and the actual path system by modifying the weight of the links (by procedure AWS), and repeat this step until the difference decreases.
   (c) There can be paths correlating to each other, so they cannot be corrected one after the other in Step 2b. Correct these paths (by procedure SWS) until the difference between the predefined and the actual path system decreases.
   (d) If all repairable errors are corrected, then STOP. Else go to Step 2a.

If it is required to decompose the circles, then the same backup path system cannot be reproduced as in the predefined case. In most of the cases, however, the overall performance of the network still remains the same after the decomposition.

Based on the computational results presented in Section 6, we can say that our algorithm practically can solve the problem of mapping dedicated paths into OSPF-based paths.

The following subsections describe the above-mentioned procedures applied by the OWS algorithm.

3.1. Initialization

First of all, initial link weights are given (see procedure IWS in Section 3.1.1) and the existence of possible circles of backup links is analyzed and the decomposition of them is done. First we show that no weight system can reproduce the predefined path system in case of circles (see Section 3.1.2), however, in most of the cases, the overall performance of the network can remain the same by the decomposition strategies shown for the two possible configurations of circles (see procedure DEC_1 in Section 3.1.3 and DEC_2 in Section 3.1.4).

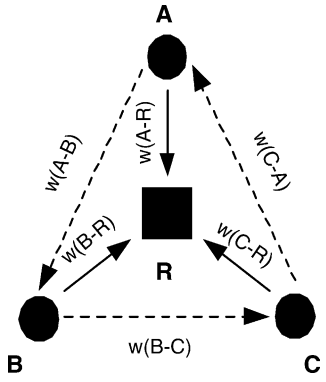

Fig. 2. Directed circle of backup links.

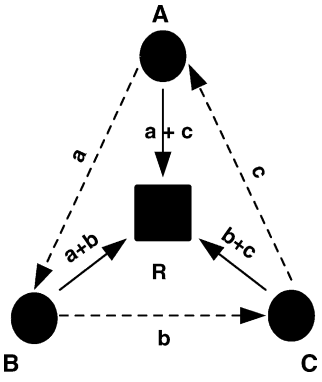

Fig. 3. Directed circle with merging default paths at the same node.

3.1.1. Initial weight setting (IWS)

According to the conditions, the traffic paths must use only default links in case of regular network operation and each backup path must contain only one backup link. In order to meet these conditions, we set the weight of each default link to 1 and set the weight of the backup links to 3. However, this weight setting does not provide that the traffic between two nodes actually uses its predefined paths. Therefore some links may become overloaded, so we need further refinement in the weight setting.

3.1.2. Directed circles of backup links

Claim 1. *A circle of backup links causes an infeasible situation if they are on the same level in the network and form a directed circle according to the assumptions in Section 2. Fig. 2 illustrates such a circle.*

Proof. For the sake of simplicity, consider a circle of three links. Let us suppose to the contrary that a proper weight system exists. This weight system the following conditions are held:

$$\begin{aligned} W(A-B)+W(B-R)&<W(C-A)+W(C-R)\\ W(B-C)+W(C-R)&<W(A-B)+W(A-R)\\ W(C-A)+W(A-R)&<W(B-C)+W(B-R) \end{aligned} \tag{1}$$

Summing up these in inequalities, we get the same on both sides, which is a contradiction. □

That means that there are no proper weight systems to map the dedicated routing. (The proof is similar for the case of circles with more nodes.) However, the overall performance of the network reliability can be preserved if we enable some modifications in the predefined path system. In the next sections, these techniques are presented. For the sake of simplicity, the circles of the examples consist of three links. Of course, both techniques can be extended to circles of more links.

3.1.3. Circles with merging default paths at the same node (DEC_1)

According to the above proof, we cannot solve the directed circle problem, but if the default paths of the nodes in the circle merge in the same node, then we can *decompose* the circle. This means that by modifying the backup path system we can delete a link from the circle without degrading the performance of the network. Fig. 3 illustrates the case when the default paths merge in the root. (Now let letters a, b and c denote the traffic demand of the nodes, thus the required capacity of the links, as well.)

Let c be the smallest traffic demand (or one of them if there are more). If there is a failure between nodes C and R, then the traffic c can go also towards node B. Since we assume one-failure scenario, this traffic c can also go through node B towards node R, because it is not greater than traffic a (there is backup path on link B–R for node A, whose default path still works). These altogether mean that the backup link C–A is needless. So we can delete this backup link, use link B–C as backup link in both direction and solve the problem with procedure *AWS*. So the overall performance of the network remains the same.

3.1.4. Circles with default paths merging at different nodes (DEC_2)

Fig. 4 illustrates the situation, when some default links of the nodes in the circle merge before all of them are merged: default paths of nodes A and C are merged at node D, but the default path of node B is merged to them only at node R.

The result of the decomposition depends on the amount of the traffic of the nodes:

- *Success:* If node B or C has the smallest traffic, then the circle can be decomposed as in Section 3.1.3.
- *Failure:* If node A has the smallest traffic, then only its traffic can be rerouted. Since C is not so protected as A (node C is not protected for failure of link D–R, but A is), node A would not have backup path for all failures protected by the predefined case. So the circle cannot be decomposed and the planning problem cannot be solved.

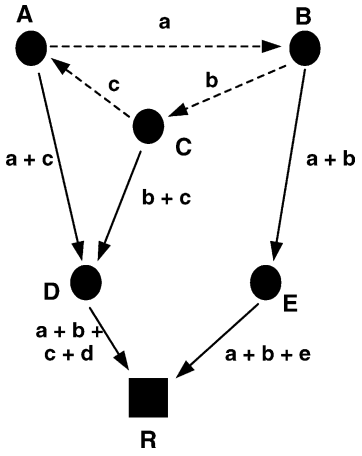

Fig. 4. Directed circle with default paths merging at different nodes.

3.2. Iterative weight adjustment

In the second part of the OWS algorithm, an iterative weight adjustment is proposed. First the link overload is decreased in the network (see procedure BWS in Section 3.2.1). Then the difference between the predefined and the actual path system is decreased (see procedure AWS in Section 3.2.2). After that those paths are corrected, which correlate to each other (see procedure SWS in Section 3.2.3). Finally these three steps are repeated until all repairable errors are corrected.

3.2.1. Basic weight setting (BWS)

We present a simple procedure (called *BWS*) that tries to find a weight system which provides that there are no overloaded links in the network.[1] The framework of the procedure is the following:

1. Calculate the total network overload (the sum of the overload on the links; denoted by O_{old}).
2. For all $v \in V$ do:
   (a) Simulate each failure scenario i for $p_v^i \neq \varnothing$. If a link is overloaded in any of the scenarios, then increase its weight by 1.
   (b) Repeat Step 2a until there is an overloaded link in case of the failures scenarios.
3. Calculate again the total network overload (denoted by O_{new}).
4. If $O_{\text{new}} < O_{\text{old}}$, then go to Step 2, else stop.

3.2.2. Advanced weight setting (AWS)

We propose a procedure taking the default–backup system of our model into consideration and decreasing the difference between the predefined and the actual path system. This procedure can correct almost all overload situations, which may arise in the network. We note that procedure *AWS* is more efficient if the initial weights of the backup links are set to a value larger than 3. We correct the weight system of the nodes by going down the topology from the root to the leaves:

1. Let $L = 1$.
2. Each failure scenario i is simulated for each $v \in V$ for which $l(v) = L$ and $p_v^i \neq \varnothing$.
   Find the illegal paths (non-backup and non-default paths) used by v. Let the weight of these paths be $W_{\mathcal{I}}$. If such an illegal path exists, then we do the following:
   (a) Find a still existing backup path for v, which has the smallest weight $W_{\mathcal{B}} := W(p_v^k)$.
   (b) Correct the weight of the first default link which can be found on each illegal path from v to r. The correction is to increase their weight by $W_{\mathcal{B}} - W_{\mathcal{I}} + 1$. (The weight of a particular corrected link is increased only once even if it is in several illegal paths.)
3. $L \leftarrow L + 1$. If $\exists v \in V$, $l(v) = L$ then go to Step 2.
4. If the difference between the predefined and the actual path system is decreased, then go to Step 1, else STOP.

Note that it is more beneficial to start the correction of the illegal paths with those nodes which have shorter illegal paths. Since the procedure modifies only the weight of the default links, it must be completed with some special cases presented in Section 3.2.3

3.2.3. Special weight setting (SWS)

In some situations, the failures can be corrected separately, but if they are combined together, they cannot be solved by procedure *AWS*. Fig. 5 shows an example for this special situation.

Node *A* has a backup path *A–F–E–D–R* and node *F* has a backup path *F–A–B–C–R*. If we wanted to correct node *A* in case of failure on link *C–R*, then we should increase the weight of link *A–B* (see the round brackets). If we wanted to correct node *F* in case of failure on link *D–R*, then we should increase the weight of link *F–E* (see the brackets). It is easy to see that the correction of nodes *A* and *F* after each other cause

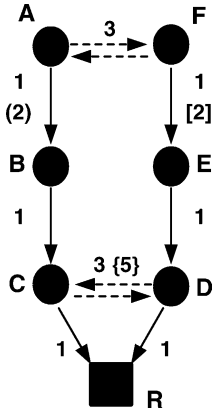

Fig. 5. Special case: special weight setting (SWS).

[1] If there is an overloaded link, then the actual path system surely differs from the predefined one.

an infinite cycle for procedure *AWS* (without the termination condition). Therefore, if we want to correct both nodes *A* and *F*, then we have to modify links *C–D* and *D–C* (see the braces). The new weights for these links are as Eq. (2):

$$w(C-D) = w(D-C) = \max(w(A-F) + w(F-E) + w(E-D), w(F-A) + w(A-B) + w(B-C)) \quad (2)$$

Of course, this procedure can be easily extended to the case, when there are other nodes between nodes *B* and *C* and between *E* and *D*.

4. Reference solutions

In this section, we present a randomized weight setting algorithm [6] used as reference method in the evaluation of our proposed weight setting procedure. This algorithm is based on the well-known Simulated Annealing [7,8] metaheuristic, which is an efficient technique for solving complex optimization tasks with large state space. The most important steps of the reference algorithm are listed below:

- *Initialization.* The *initial weight setting* method (presented in Section 3) is used to adjust the initial weights for the default and backup links.
- *Demand allocation.* According to the current weight system all demands are routed and all failures are simulated. Then the required link capacities are calculated and the overloaded links are searched, as well as the total network overload is calculated (denoted by O_{old}).
- *Weight modification.* In this phase we can select from two possibilities, the first is a simple weight adjustment, and the second one is a sophisticated strategy.
  1. One overloaded link is selected randomly and its weight is incremented by 1.
  2. Select a link randomly and modify its weight by 1 according to the probability P_{mod}: if the link is overloaded, then its weight should be increased and P_{mod} = overload/maximal overload; if the link is underutilized, then its weight should be decreased and P_{mod} = underutilization/maximal underutilization. If we do not modify weight of the link, then select another link randomly.

     Then all failures are simulated, the overloaded links are searched and the total network overload is calculated again (denoted by O_{new}).
- *Evaluation.* Using the so-called stochastic acceptance criteria of simulated annealing it is decided whether the modification of the weight is acceptable or not. The network modification will be accepted with the probability

$$P_{\text{accept}} = \min\left\{1, \exp\left(-\frac{O_{\text{new}} - O_{\text{old}}}{T}\right)\right\} \quad (3)$$

where T is the so-called temperature, which decreases exponentially during the running of the algorithm. If the overload in case of the new weights is lower than in case of the weights before the modification the new weight system is always accepted. If the new overload is greater than the old one, then the acceptance depends on the above criteria. At the beginning of the optimization the probability of the acceptance of an overloaded state is close to 1, while later this probability decreases significantly.

After the decision, the optimization continues at the *Demand allocation* step.

5. Exact solution

This section exhibits a linear programming based exact solution to the problem of finding OSPF weight setting with various objective functions.

To give an LP formalization of the problem we use the following well-known duality theorem of shortest paths.

Let $G=(V, E)$ is a directed graph, and $w : E \to \mathbb{Z}_+$ a weight function on the edges. A $\pi : V \to \mathbb{Z}$ potential is called *feasible* if

$$w_{uv} \geq \pi(v) - \pi(u) \quad (4)$$

holds for all edges $(uv) \in E$. An edge uv is called π-*tight* or *tight* if $w_{uv} = \pi(v) - \pi(u)$.

Claim 2. *For any fixed node $s \in V$ there exists a feasible potential π, such that a path p from s to an arbitrary node t is a shortest s–t path if and only if each edge of p is π-tight.*

Proof. Let us define $\pi(v)$ to be the length of the shortest s–v path. It is easy to check that this potential is feasible and meets the requirements above. □

The following claim is also easy to see.

Claim 3. *If a path p is not a shortest path, then there exists no feasible potential π such that each edge of p is π-tight.*

Using these claims, we are able to formulate our problem. Our goal is to find a weight function of the edges, by which the shortest path of the nodes equal to their predefined paths in all states of the network. These states are the normal work state (0) and the states of the possible default link failures (1,…,k, where k is the number of nodes). To sum up, we are looking for such weight system, by which the potentials are feasible in all states and all the links used in ith state (referred to as E^i) are π-tight and for all other links $w_{uv} > \pi^i(v) - \pi^i(u)$ holds (where $\pi^i(u)$ is

the potential of node u in the ith state). The linear program formulates as follows.

For each edge (u, v) a variable $w(u, v)$ expressing the OSPF weight is introduced. Moreover for each fault scenario i (0 for the normal work state and 1,…,k, for all possible default link failures) and for each node u we introduce a variable $\pi^i(u)$

$$w(u,v) \geq 1 \quad \forall\, (u,v) \in E \tag{5a}$$

$$\pi^i(u) \geq 0 \quad \forall\, u \in V, i = 1,\ldots,k \tag{5b}$$

$$w(u,v) = \pi^i(v) - \pi^i(u) \quad \forall\, (u,v) \in E^i \tag{5c}$$

$$w(u,v) \geq \pi^i(u) - \pi^i(v) + 1 \quad \forall\, (u,v) \in E^i \tag{5d}$$

$$w(u,v) \geq \pi^i(v) - \pi^i(u) + 1 \quad \forall\, (u,v) \in E \backslash E^i \tag{5e}$$

$$w(u,v) \geq \pi^i(u) - \pi^i(v) + 1 \quad \forall\, (u,v) \in E \backslash E^i \tag{5f}$$

Claim 4. *The above linear program is feasible if and only if there exists an appropriate weightings.*

Proof. The above linear inequalities express that π^i is a feasible potential in the fault scenario i and exactly the elements of E^i are tight. According to Claims 2 and 3 this means that the required set of paths are unique shortest path with respect to the weighting $w(u, v)$.

On the other hand, if $w(u, v)$ is a proper weightings then let $\pi^i(u)$ be the distance of the node u from the root in the ith fault scenario. This gives a feasible solution of the above linear program. □

If our aim is to find solution with 'small' weights, i.e. we want to minimize the largest weight, then we can introduce an additional auxiliary variable Z and minimize it keeping the above constraint and also the following

$$Z \geq w(u,v) \quad \forall\, (u,v) \in E \tag{5g}$$

6. Computational results

First we compare the running time of our *OSPF Weight Setting—OWS* algorithm and the LP solution given by the LP-solver package [9].

Table 1 shows that *OWS* can solve large problem instances in approximately half a minute, however, *ILP* cannot reach solution even for 100 nodes in 3 days. Therefore we compare the results of the *OWS* algorithm with the two simulated annealing based *Reference algorithms* (*RefSol-1*, *RefSol-2*) and the *Initial Weight Setting—IWS* procedure.

Table 1
Average running times

Algorithms	Number of nodes			
	50	100	150	200
OWS (s)	4.16	8.03	18.6	35
ILP	≈1 h	Uncertain	Uncertain	Uncertain

Because the goal is to test the OWS in case of real network sizes, we use networks with 50, 100, 150, 200 nodes. Ten networks of each size are generated, the topology of the networks satisfies all the technological requirements and constraints (maximum 2 incoming links per node, the depth of a tree is 5) and it is very close to an optimized UMTS access network. Furthermore, there were no circles from backup links (see Section 3.1.2) in any instance networks. We have no exact values of what percentage of the RBSs are planned to be protected in the access network, therefore we examine four different protection level scenarios. In case of each network size, we examine the cases when 10, 20, 30 and 40% of the RBSs have a backup link. (Note that above these backup link values ring or mesh-like topologies would be more efficient to be applied instead of the extension of trees.)

Because of both the *OWS* algorithm and the *IWS* procedure are deterministic, only one run is enough to obtain their results. However, the *Reference algorithms* are based on randomized procedure, which means that more than one run is required to obtain a typical, average result. Thus we ran the *Reference algorithms* 10 times for each network sizes and protection level.

We compare the methods mentioned above according to percent of lost traffic (value of overload in the network), and percent of number of overloaded links. The results are summarized in Figs. 6 and 7, respectively.

Fig. 6 shows that the *IWS* procedure does not solve the OSPF weight setting problem and the overload in the network increases as the number of backup links increases. When 10% of the links are backup link, the *RefSol-1* can solve the problem for all network sizes, however, *RefSol-2* can solve the problem in case of smaller networks (50 and 100 nodes). Moreover, in case of more backup links, the overload increases, but it still remains under 0.6%. This result is acceptable compared to the *IWS* procedure, nevertheless, the *OWS* algorithm can *always* solve the problem.

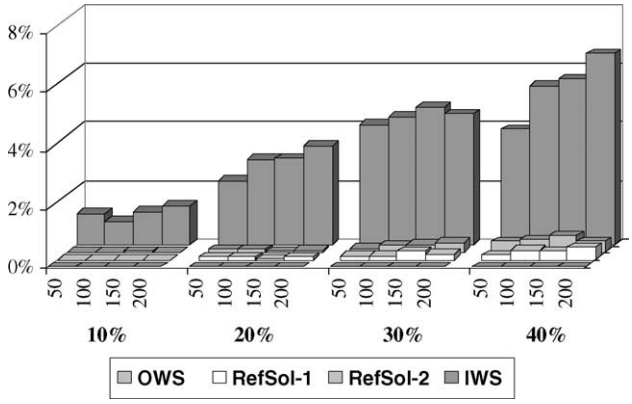

Fig. 6. Average overload in percent.

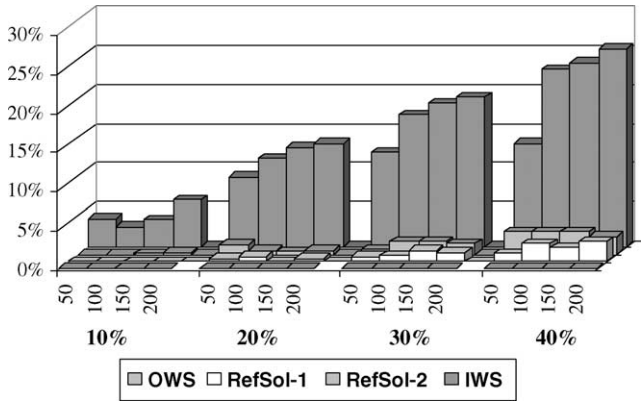

Fig. 7. Average number of overloaded links in percent.

Fig. 7 shows that the number of overloaded links has a very similar behavior to the overload. In case of the *IWS* procedure the number of overloaded links mostly depends on the number of backup links. In case of the *Reference solutions*, the number of overloaded links slightly increases as the number of backup links increases. Moreover, the number of overloaded links is the multiple of the overload in the network. The reason of this is that the *Reference solutions* try to minimize the overload in the links, so they prefer such solutions, where the overload is shared between several links. Of course, in case of the *OWS* algorithm, there are no overloaded links in the network.

The above tests back up that our novel *OWS* algorithm can solve the OSPF weight setting problem (or at least keep the performance of the network on the same level as in the dedicated case if there are directed circles in the network).

7. Conclusions

In this paper we dealt with the configuration of OSPF weight system in protected UMTS access networks. The goal was to propose a weight setting strategy that makes the default path to be the shortest path in the network in case of nominal operation and the backup paths to be the shortest path in the network in case of any single link failure. We showed that the simple, greedy type weight setting fails to solve this task, so it results very significant overload and traffic loss, so it is not applicable. The known weight settings result in measurably better network configuration, but some overload still remains. Therefore a more sophisticated method has been worked which (a) considers the special properties and structure of UMTS access networks, as well as (b) finds those parts of the explicit path structure, which does not conform with the OSPF routing rule. Based on some tests, we have proven that our proposed method is able to set the OSPF weight in a proper way in cases of different network topologies and different number of protected demands. The results shows that our method is able to solve the OSPF weight setting task in real UMTS/OSPF access networks and it is an efficient, useful tool in the network planning process.

References

[1] T. Ojanperä, R. Prasad, Wideband CDMA for Third Generation Mobile Communication, Artech House Publishers, 1998.
[2] S. Dravida, H. Jiang, M. Kodialam, B. Samadi, Y. Wang, Narrowband and broadband infrastructure design for wireless networks, IEEE Communications Magazine May (1998).
[3] OSPF Version 2. RFC2328, www.ietf.org/rfc/rfc2328.txt, 1998.
[4] B. Fortz, M. Thorup, Internet traffic engineering by optimizing OSPF weights, Proceedings of the INFOCOM 2000, Tel-Aviv 2000.
[5] A. Szlovencsák, I. Gódor, J. Harmatos, T. Cinkler, Planning reliable UMTS terrestrial access networks, IEEE Communications Magazine 40 (1) (2002) 66–72.
[6] M. Pióro, Á. Szentesi, J. Harmatos, A. Jüttner, S. Kozdrowski, On OSPF related networks optimization problems, Eighth IFIP Workshop on Performance Modelling Evaluation of ATM and IP Networks, Ilkley, UK, July 17–19, 2000.
[7] B. Hajek, A tutorial survey of theory and applications of simulated annealing, Proceedings of the 24th Conference on Decision and Control, Ft. Lauderdale, FL, December 1985.
[8] B. Hajek, Cooling schedules for optimal annealing, Mathematics of Operation Research 13 (2) (1988).
[9] M. Berkelaar, J. Dirks, lp_solve 2.2, ftp://ftp.es.ele.tue.nl/pub/lp_solve.