Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

ANDREAS BLEY AND THORSTEN KOCH

Integer programming approaches to access and backbone IP-network planning

URL: http://www.zib.de/projects/telecommunication/

ZIB-Report 02-41 (2002)

Integer programming approaches to access and backbone IP-network planning*

Andreas Bley and Thorsten Koch

November 25, 2002

Abstract

In this article we study the problem of designing a nation-wide communication network. Such networks usually consist of an access layer, a backbone layer, and maybe several intermediate layers. The nodes of each layer must be connected to those of the next layer in a tree-like fashion. The backbone layer has to satisfy certain survivability and routing constraints.

Given the node locations, the demands between them, the possible connections and hardware configurations, and various other technical and administrational constraints, the goal is to decide, which node is assigned to which network level, how the nodes are connected, what hardware must be installed, and how traffic is routed in the backbone.

Mixed integer linear programming models and solution methods are presented for both the access and the backbone network design problem. The focus is on the design of IP-over-SDH networks, but the access network design model and large parts of the backbone network design models are general and also applicable for other types of communication networks. Results obtained with these methods in the planning of the German research network are presented.

Keywords: Network design, Traffic engineering, Internet routing, Mixed-integer programming

Mathematical Subject Classification (2000): 90B18, 90C11, 90C27, 90C90

1 Introduction

The German gigabit research network G-WiN, operated by the DFN-Verein e.V., is the largest IP network in Germany. In this article we describe the mathematical models and tools used to plan the layout and dimensioning of the access and backbone network [BK00]. Since these models are general in nature, they were used also in two other projects. One was the placement of switching centers in circuit switched networks, a cooperation with Telekom Austria. In the other project, studies about MSC planning in mobile phone networks were conducted together with E-Plus. Unfortunately, the data from the later projects cannot be published. For this reason, we focus throughout this article on the G-WiN IP network as an example, but show the generalized models that were developed.

The problem we were faced with, roughly can be stated as follows: Given the node locations, the demands between them, the possible node layers, connections, and hardware configurations, and various other technical and administrational constraints, the goal is to decide, which node is assigned to which network layer, how the nodes are connected, what hardware must be installed, and how traffic is routed in the backbone network.

*This work was partially funded by the Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF).

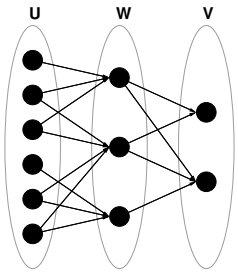

Figure 1: Network layers

In this article we present a two-phase approach that splits between access and backbone network planning. In a first step, we only consider the access network. Then, in the secound phase, the backbone dimensioning and routing problem is addressed. Both problems are solved by integer linear programming techniques.

The problems encountered in the access network planning can be viewed as capacitated two-level facility location problems. There is a huge amount of literature on various aspects of this kind of problems. See for example [MF90, ALLQ96, Hal96, PLPL00, MD01]. The backbone network planning problem is a capacitated survivable network design problem with some additional constraints to make sure that the routing can be realized in practice with the OSPF routing protocol, which is the dominating routing protocol in the internet. The general capacitated survivable network design problem is well studied in the literature, see for example [GMS95, AGW97, BCGT98]. A lagrangean approach to compute OSPF routing weights that minimize the maximum link utilization was proposed in [LW93], several heuristic algorithms for this problem can be found in [ERP01, BKSTG00, FT00, FGL$^+$00]. A model and heuristics to solve an integrated network design and OSPF routing problem are discussed in [BGW98]. In [BAG00] properties of shortest path routings were studied and a linear programming appoach to compute routing weights for a given shortest path configuration was proposed. Polyhedral approaches to the network design problem with shortest path routing for unicast and multicast traffic were presented in [HD01] and [Pry02], respectively.

This article is organized as follows: In Section 2 we describe the problem setting in more detail. In Section 3 we present the mathematical model for the access network planning problem. The same is done for the backbone network in Section 4. In Section 5 we describe the algorithms used and in the last section we report on computational results.

2 Problem description

The network design and routing problem studied in this article can be described as follows. Given are three node sets, a set U of demand nodes (locations), a set W of possible intermediate nodes, and a set V of possible backbone nodes (see Figure 1). The same location must belong to each set U, W, and V, if it may be assigned to each of the corresponding layers. Additional to the node sets U, W, and V, we have sets A_{UW} and A_{WV} with all potential connections between nodes from U and W, and W and V, respectively. For each pair of demand nodes $u, v \in U$ the directed traffic demand is $d^{u,v} \in \mathbb{R}^+$.

Various hardware or technology can be installed or rented at the nodes or potential edges of the network. This hardware has a modular structure and combined hardware components must have matching interfaces. For example, a router provides several slots, these can be equipped with cards providing link-interfaces, and, in order to set up a certain link-type on an edge, corresponding link-interfaces are needed at the two terminal nodes. The set of all available component types is denoted by C. We distinguish between node components C_V, edge components C_E, or global components C_G, depending on whether they can be installed at nodes, on edges, or globally for the entire network. Components may be real, like router cards or leased lines, or artificial, like bundles of components that can be purchased only together.

A component, if installed, provides or consumes several resources. The set of all considered resources is denoted by R. For each node component $c \in C_V$, resource $r \in R$ and node $v \in V$ the number $k_v^{c,r} \in \mathbb{R}$ denotes how much of resource r is provided (if > 0) or consumed (if < 0) by component c if installed at node v. Analogously, $k_e^{c,r}$ and $k_G^{c,r}$ specify the provision or consumption of for edge and global components if installed. We distinguish between three classes of resources, node resources R_V, edge resources R_E, and global resources R_G. For each edge resource r and each edge e, the total consumption must not exceed the total provision of resource r by the components installed on edge e and its incident vertices. Analogously, the consumption must not exceed the provision by the components installed at a node and its incident edges for node resources or by the components installed on all edges, nodes, and globally for global resources. The component cost and the routing capacity are special resources.

Due to the forest structure of the solutions, for the access network planning problem it is often possible to simplify the possible hardware combinations to a set of relatively few configurations and reassign node costs to edges in the preprocessing. Also the directed traffic demands can be aggregated. Hence, we can assume that for the access network planning we are given a set of assembly stages S_n, for each node n from W or V, and that κ_n^{rs} describes the capacity of resource r that will be provided at node n if it is at stage s. Regarding the traffic demands as a special resource, each location $u \in U$ has an undirected demand of δ_u^r for each resource r. For each v node from W and V the assembly stage has to be chosen in such a way, that there is enough capacity to route the demands of all resources for all demand nodes that are connected to this node v.

The backbone nodes and edges have to be dimensioned in such a way, that the accumulated demands can be routed according to the OSPF specification. Since in our approach the access network planning problem is solved first, the set of backbone nodes is given and fixed for the backbone network planning problem. For notational convenience, it will again be denoted by V. The set of potential links between these nodes is denoted by E, the graph $G = (V, E)$ is called supply graph.

The traffic demands between the demand nodes U are aggregated to a set of demands between the backbone nodes V. We assume that between each pair of nodes $v_1, v_2 \in V$ there is a demand (maybe equal to zero) denoted by d^{v_1,v_2}.

We wish to design a backbone network that is still operational if a single edge or node fails. Therefore, we introduce the set of operating states $O \subseteq V \cup E \cup \{\emptyset\}$. We distinguish between the normal operating state $o = \emptyset$, which is the state with all nodes and edges operational, and failure states, which are the states with a single edge ($o = e \in E$) or a single node ($o = v \in V$) non-operational. Note that in a node failure state $o = v \in V$ all edges incident to v are non-operational, too, and the demands with origin or destination v cannot be satisfied and are therefore not considered in this operating state.

Unfortunately, in the backbone network planning we cannot simplify the hardware combinations like in the access network planning. Since we always had significant costs associated with or several restrictions on the use of certain hardware components, it was necessary to explicitly consider all single hardware components.

The capacities provided by the components installed on the edges must be large enough to allow a feasible routing with respect to the OSPF routing protocol in all operating states. Assuming non-negative routing weights for all arcs, the OSPF protocol implies that each demand is sent from its origin to its destination along a shortest path with respect to these weights. In each operating state only operational nodes and arcs are considered in the shortest path computation. In this article we address only static OSPF routing where the weights do not depend on nor change with the traffic. Dynamic shortest path routing algorithms, which try to adapt to traffic changes, often cause oscillations that lead to significant performance degradation, especially if the network is heavily loaded (see [CW92]). Also, though most modern IP routers support OSPF extensions that allow to split traffic onto more than one forwarding arcs, in this article we consider only the standard non-bifurcated OSPF routing. This implies, that the routing weights must be chosen in such a way that, for all operating states and all demands, the shortest path from the demand's origin to its destination is unique with respect to these weights. Otherwise, it is not determined which one of

the shortest paths will be selected by the implementation of the routing protocol in the real network and, therefore, it would be impossible to guarantee that the chosen capacities permit a feasible routing of the demands.

The variation of data package transmission times increases significantly with the number of nodes in the routing path, especially if the network is heavily loaded. In order to guarantee good quality of service, especially for modern real-time multi-media services, the maximum routing path length is bounded by a small number (at least for the normal operating state). Also, even though OSPF routing in principle may choose different paths for the routing from s to t and t to s, a symmetric routing was preferred by the network administration.

For different planning horizons, usually also different optimization goals are used. In the long-term strategic planning the goal is to design a network with minimal cost. In the short-term operational planning the goal often is to improve the network's quality with no or only very few changes in the hardware configuration. Since Quality-of-Service in data networks is strongly related to the utilization (load or congestion) of the network, typical objectives chosen for quality optimization thus are the minimization of the total, average, or maximum utilization of the network's components. The utilization of a network's component is ratio between the traffic flow through this component and its capacity. In order to provide the best possible quality uniformly for all connections and to hedge against burstiness in traffic, a variant of minimizing the maximum utilization of the edge components was chosen as objective function for operational backbone network planning in our application.

Usually, simply minimizing the maximum utilization over all network components in all operating states makes no sense in practice. Often there are some components or links in the network that always attain the maximum utilization, like, for example, gateways to other networks with fixed small capacities. Also, it is not reasonable to pay the same attention to minimizing the utilization of some link in a network failure state as in the normal operating state. To overcome these difficulties, we introduce a set of disjoint load groups $j \in J$, $j \subset C \times E \times O$. Each load group defines a set of edge components on some edges in some operating states,e.g., all edge components on all interior edges in the normal operating state, and each triple (edge component, edge, operating state) belongs to at most one load group. For each load group the maximum utilization is taken only over the components, edges, and operating states in that load group. The objective is to minimize a linear combination of the maximum utilizations for all load groups. Thus, different classes of network components can be treated independently and differently in the utilization minimization. The concept of load groups can be generalized straightforward to include all node, edge, and global components. In our application, we only consider edge components in load groups.

In the next two sections, we will develop mathematical models for the problems described above.

3 Access network planning

For the access network we assume that all traffic is routed through the backbone. This leads to a forrest structure with the demand nodes as leafes and the backbone nodes as roots. One advantage of this approach is that we can map all node attributes like cost and capacity on the edges, since each node has at most one outgoing edge. In the uncapacitated case this results in a Steiner tree problem in a layered graph, which could be solved at least heuristically quite easy, see for example [KM98].

For each possible connection in A_{UW} we introduce a binary variable x_{uw}, which is 1 iff u and w are connected. Analogously, variables x_{wv} are introduced for A_{WV}. For each possible connection in A_{WV} and each resource $r \in R$ we use a continous variable f^r_{wv} that is equal to the flow of resource r between w and v . For each node n from W and V and each assembly stage $s \in S_n$ we use a binary variable z^s_n that is 1 iff stage s is selected for node n. There are no flow variables and costs between the demand and intermediate nodes, because these costs can be computed in advance and added to the installation cost for the link.

3.1 Constraints

Here we present the constraints needed to construct a feasible solution. Each demand node has to be connected to exactly one intermediate node and each intermediate node can be connected to at most one backbone node:

$$\sum_{(u,w)\in A_{UW}} x_{uw} = 1 \quad \forall u \in U \qquad \sum_{(w,v)\in A_{WV}} x_{wv} \leq 1 \quad \forall v \in W \tag{1}$$

Each intermediate and each backbone node has exactly one configuration:

$$\sum_{s\in S_w} z_w^s = 1 \quad \forall w \in W \qquad \sum_{s\in S_v} z_v^s = 1 \quad \forall v \in V \tag{2}$$

If it is possible, that node $n \in W \cup V$ is not chosen at all, there has to be a configuration with $\kappa_n^{rs} = 0$ for all $r \in R$. In the case of a planning with no current installations the cost of the corresponding variable z_n^s would be zero. Otherwise if there is already an installation, we can give this configuration zero cost and associate with all other configurations either negative costs, which means we earn something by switching to another configuration, or positive changing costs. This way, we could also add a fixed charge just for changing what is already there.

The configuration of the intermediate nodes must provide enough resources to meet the demands:

$$\sum_{(u,w)\in A_{UW}} \delta_u^r x_{uw} - \sum_{s\in S_w} \kappa_w^{rs} z_w^s \leq 0 \qquad \forall w \in W, r \in R \tag{3}$$

Demands can only be routed on chosen connections:

$$\lambda_w^r x_{wv} - f_{wv}^r \geq 0 \quad \forall (w,v) \in A_{WV}, r \in R \quad \text{and} \quad \lambda_w^r = \max_{s\in S_w} \kappa_w^{rs} \tag{4}$$

The flow balance at each intermediate node has to be ensured:

$$\sum_{(u,w)\in A_{UW}} \delta_u^r x_{uw} - \sum_{(w,v)\in A_{WV}} f_{wv}^r = 0 \qquad \forall w \in W, r \in R \tag{5}$$

Equation (5) can easily be extended with a constant additive factor in case the intermediate node is serving a fixed amount of the demand. Also, a scaling factor can be incorporated if, for example, data compression is employed for the traffic from the intermediate to the backbone nodes.

The configuration of a backbone node has to provide enough resources to meed the demands of the assigned intermediate nodes:

$$\sum_{(w,v)\in A_{WV}} f_{wv}^r - \sum_{s\in S_w} \kappa_v^{rs} z_v^s \leq 0 \qquad \forall v \in V, r \in R \tag{6}$$

3.2 Objective function

While minimizing cost was always the objective, getting sensible cost coefficient for the objective function proved to be a major problem in all projects.

The objective can be a combination of the following terms:

Installation cost for connections

$$\sum_{(u,w)\in A_{UW}} \tau_{uw} x_{uw} + \sum_{(w,v)\in A_{WV}} \tau_{wv} x_{wv}, \tag{7}$$

installation cost for configurations

$$\sum_{w \in W} \sum_{s \in S_w} \tau_w^s z_w^s + \sum_{v \in V} \sum_{s \in S_v} \tau_v^s z_v^s, \tag{8}$$

and flow unit cost between intermediate and backbone nodes

$$\sum_{r \in R} \sum_{(w,v) \in A_{WV}} \tau_{wv}^r f_{wv}^r. \tag{9}$$

The $\tau_{uw}, \tau_{wv}, \tau_w^s, \tau_v^s$, and $\tau_{wv}^r \in \mathbb{R}$ are the cost coefficients used to weight the desired terms.

3.3 Notes

If a demand node can be connected directly to a backbone node, this can be modeled by introducing an additional artificial intermediate node with only one link to the backbone node and adding a constraint that forbids a "zero" configuration of the backbone if this link is active.

The model can be easily extended to an arbitrary number of intermediate levels, but, of course, solving it will become more and more difficult.

A major drawback of this model is the inability to cope with locally routed demands. If two demand nodes are connected to the same intermediate node, often the demand need not to be routed to the backbone. Finding optimal partitions that minimize the traffic in the backbone is NP-hard itself and difficult to solve [FMdS+96]. On the other hand, demands are usually quite unstable. While a "big" node will always have a high demand, the destination of its emanating traffic might change frequently. In the case of the G-WiN, a large fraction of the traffic is leaving it or entering from outside the network. We did limited experiments which indicated that, at least in this case, the possible gain from incorporating local traffic is much less than the uncertainty of the input data.

4 Backbone network planning

In this section we present a mixed-integer linear programming model for the backbone network dimensioning and OSPF routing problem.

4.1 Network hardware

In contrast to the access network, the possible configurations of the modular hardware for the backbone network are described by explicit single component variables and resource constraints. The number of installations of components are modeled by integer variables $z_v^c \in \mathbb{N}$, $\underline{z_v^c} \leq z_v^c \leq \overline{z_v^c}$, for each node component $c \in C_V$ and node $v \in V$, $z_e^c \in \mathbb{N}$, $\underline{z_e^c} \leq z_e^c \leq \overline{z_e^c}$, for each edge component $c \in C_E$ and edge $e \in E$, and $z_G^c \in \mathbb{N}$, $\underline{z_G^c} \leq z_G^c \leq \overline{z_G^c}$, for each global component $c \in C_G$. The lower and upper bounds $\underline{z_*^c}$ and $\overline{z_*^c}$ for each component and node, edge, or the global network restrict how often this component can be installed there.

Depending on the resource type, with each resource there are one or more inequalities associated. For node resource constraint we have

$$\sum_{c \in C_E} \sum_{e \in \delta(v)} k_e^{c,r} z_e^c + \sum_{c \in C_V} k_v^{c,r} z_v^c \geq 0 \quad \forall\, r \in R_V, v \in V. \tag{10}$$

Analogously, the inequalities for edge and global resource constraints are

$$\sum_{c \in C_E} k_e^{c,r} z_e^c \sum_{c \in C_V} (k_v^{c,r} z_v^c + k_w^{c,r} z_w^c) \geq 0 \quad \forall\, r \in R_E,\, vw = e \in E, \text{ and} \tag{11}$$

$$\sum_{c \in C_E} \sum_{e \in E} k_e^{c,r} z_e^c + \sum_{c \in C_V} \sum_{v \in V} k_v^{c,r} z_v^c + \sum_{c \in C_G} k_G^{c,r} z_G^c \geq 0 \quad \forall\, r \in R_G. \tag{12}$$

The node configurations $s \in S_w$ chosen for the backbone nodes in the access network planning phase can be regarded as special fixed components now, which provide the resources to install further components. The optimization goal of minimizing the total network cost can be easily formulated as

$$min \sum_{c \in C_E} \sum_{e \in E} k_e^{c,cost} z_e^c + \sum_{c \in C_V} \sum_{v \in V} k_v^{c,cost} z_v^c + \sum_{c \in C_G} k_G^{c,cost} z_G^c, \tag{13}$$

where $cost \in R_G$ is the special cost resource. Of course, any other objective that is linear in the components can be formulated as well via an appropriate resource.

4.2 Routing

There are several possible ways to model the OSPF routing of the demands. We used arc–flow, path–flow, and tree–based formulations in our practical experiments. In this article, we present a formulation that is based on binary arc–flow variables.

In order to model the directed OSPF traffic appropriately, with each edge $e = uv \in E$ we associate the two directed arcs (u, v) and (v, u) and let $A = \{(u, v), (v, u) \mid uv \in E\}$. To simplify the notation, we use V^o, E^o, and A^o to denote the sets of operational nodes, edges, and arcs in operating state o, respectively.

For each operating state $o \in O$ and each traffic demand we use a standard formulation with binary arc–flow variables, i.e., there is a variable $p_a^{s,t,o} \in \{0, 1\}$ for all $o \in O$, $s, t \in V^o$, $a \in A^o$, $s \neq t$, with $p_a^{s,t,o} = 1$ iff arc a is in the routing path from s to t is operating state o. The flow balance and edge capacity constraints then are

$$\sum_{a \in \delta^+_{A^o}(u)} p_a^{s,t,o} - \sum_{a \in \delta^-_{A^o}(u)} p_a^{s,t,o} = \begin{cases} 1 & u = s \\ -1 & u = t \\ 0 & \text{otherwise} \end{cases} \quad \forall\, o \in O,\, s, t, u \in V^o, \text{ and} \tag{14}$$

$$\sum_{s,t \in V^o} d^{s,t} \cdot p_{(u,v)}^{s,t,o} \leq \sum_{c \in C_E} k_{uv}^{c,cap} z_{uv}^c \quad \forall\, o \in O,\, (u, v) \in A^o. \tag{15}$$

In our application, there were no node capacity constraints necessary to model, but they can be included into the model in a straightforward way. The allow only symmetric routing, the inequalities

$$p_{(u,v)}^{s,t,o} = p_{(v,u)}^{t,s,o} \quad \forall\, o \in O,\, s, t \in V^o,\, (u, v) \in A^o \tag{16}$$

can be added to the formulation. The maximum routing path length is enforced by the inequalities

$$\sum_{a \in A^o} p_a^{s,t,o} \leq \bar{p}^{s,t,} \quad o \in O,\, s, t \in V^o, \tag{17}$$

where $\bar{p}^{s,t,o}$ is the maximum path length allowed between s and t in operating state o.

4.3 OSPF Routing

So far, we presented a mixed-integer linear programming model for a general survivable network design problem with single-path routing. In this section, we will show how to incorporate the special features of the OSPF routing protocol into this model.

It is easy to see, that not all possible configurations of routing paths are realizable with a shortest path routing protocol. For some path configurations it is impossible to find weights, such that all paths are shortest paths simultaneously. Many, rather complicated constraints have to be satisfied by a path configuration to be realizable by some routing weights. Examples of such constraints are presented, for example, in [BAGL00], [BAG00], [Gou01], and [SKK00]. We will call a path configuration admissible if it can be realized by a set of routing weights.

In the following, we will present some simple necessary constraints for admissible path configurations and what inqualities these constraints impose on the path variables p. Unfortunately, a complete description of all admissible path configurations by inequalities (in a template-scheme) on the path variables is not known.[1] But it is possible to decide in polynomial time by solving a linear program whether a given path configuration is admissible and, if not, to generate an inequality that is valid for admissible path configurations but violated by the given one. We will present this linear program and show how it can be used to separate inequalities cutting off non-admissible path configurations.

Subpath constraints

The simplest constraints that must be satisfied by the routing paths are the subpath (or path monotony) constraints. Suppose we have a set of routing weights w and $P^{s,t}$ is a unique shortest path between $s, t \in V$, $s \neq t$, of length at least 2. Let $v \in V$ be an inner node of $P^{s,t}$. Then, if $P^{s,v}$ or $P^{v,t}$ denote the shortest paths between s, v and v, t, respectively, both paths must be sub-paths of $P^{s,t}$. Otherwise, $P^{s,t}o$ cannot be the unique shortest path between s in t, see Figure 2.

Hence, the routing variables of all admissable path configurations must satisfy the subpath inequalities:

$$p_a^{s,v,o} \leq p_a^{s,t,o} + \left(1 - \sum_{a' \in \delta^-_{A^o}(v)} p_{a'}^{s,t,o}\right) \qquad \forall\, o \in O,\ s,t,v \in V^o,\ a \in A^o. \tag{18}$$

$$p_a^{v,t,o} \leq p_a^{s,t,o} + \left(1 - \sum_{a' \in \delta^-_{A^o}(v)} p_{a'}^{s,t,o}\right) \qquad \forall\, o \in O,\ s,t,v \in V^o,\ a \in A^o. \tag{19}$$

Note that, in general, the subpath inequalities are not facet defining in this basic form but can be turned into facets by appropriate lifting.[2] Nevertheless, in our branch-and-cut implementation, we only use the unlifted form of the subpath inequalities (18) and (19) together with the (still not facet-defining) simple

[1] Our polyhedral studies revealed, that—even for extremely small underlying graphs—the facial structure of the polyhedron defined by the admissible path configurations is extremely complicated. Nevertheless, large classes of valid and facet-defining inequalities for admissible path systems are known.

[2] An even weaker form of subpath inequalities was used in [SKK00] and [Pry02]:

$$p_a^{s,v,o} \leq p_a^{s,t,o} + (1 - p_{(v,t)}^{s,t,o}) \qquad \forall\, o \in O,\ (v,t), a \in A^o,\ s \in V^o.$$

$$p_a^{v,t,o} \leq p_a^{s,t,o} + (1 - p_{(s,v)}^{s,t,o}) \qquad \forall\, o \in O,\ (s,v), a \in A^o,\ t \in V^o.$$

For a path $P^{s,t,o}$ these inequalities enforce the subpath property only for the subpaths $P^{s,t',o}$ and $P^{s',t,o}$, where s' and t' are the first and the last inner nodes of $P^{s,t,o}$, respectively. By transitivity, these inequalities already imply that all subpath constraints are satisfied for any integer feasible solution p. But it is easy to verify that for fractional path variables p the unlifted form of these inequalities is strictly weaker than (18) and (19).

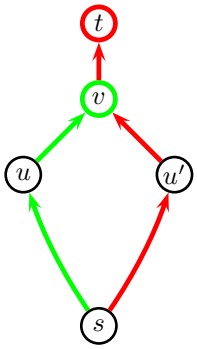

Figure 2: Example for a violated sub-path constraint

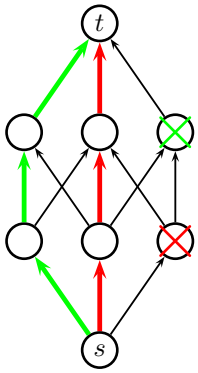

Figure 3: Example for a violated operating state coupling constraint

lifted version

$$p_a^{s,t,o} + p_a^{v,t,o} \leq p_a^{s,t,o} + 2\left(1 - \sum_{a' \in \delta^-_{A^o}(v)} p_{a'}^{s,t,o}\right) \qquad \forall\, o \in O,\ s,t,v \in V^o,\ a \in A^o. \tag{20}$$

These inequalities proved already sufficiently tight to obtain good practical results, even without the complicated lifting procedures.

Operating state coupling constraints

The operating states coupling constraints are the simplest constraints between routing paths of different operating states. Let $o_1, o_2 \in O$, $o_1 \neq o_2$, be two different operating states. Suppose for some given routing weights the paths P^{s,t,o_1} and P^{s,t,o_2} are the unique shortest paths between $s,t \in V^{o_1,o_2}$, $s \neq t$, in operating states o_1 and o_2, respectively. Furthermore, suppose that in operating state o_1 no edge or node on P^{s,t,o_2} fails and that in operating state o_2 no edge or node on P^{s,t,o_1} fails. Then both paths would remain feasible (s,t)-paths in both operating states. Since only one of them can be the shorter path, both must be identical (see Figure 3).

For notational convenience, we define the following "trigger term":

$$[o_i \in P^{s,t,o_j}] := \begin{cases} 0 & \text{if } o_i = \emptyset, \\ p_{(u,v)}^{s,t,o_j} + p_{(v,u)}^{s,t,o_j} & \text{if } o_i = uv \in E, \\ \sum_{a \in \delta^-(v) - o_j} p_a^{s,t,o_j} & \text{if } o_i = v \in V. \end{cases}$$

All admissible path configurations must satisfy the following operating state coupling inequalities:

$$p_a^{s,t,o_2} \leq p_a^{s,t,o_1} + [o_1 \in P^{s,t,o_2}] + [o_2 \in P^{s,t,o_1}] \qquad \forall\, o_1, o_2 \in O,\ s,t \in V^{o_1,o_2},\ a \in A^{o_1,o_2}. \tag{21}$$

Also the operating state coupling inequalities are not facet defining in this basic form in general, but can be turned into facets by a lifting procedure.

Computing routing weights

Let w be routing weights that induce unique shortest path between all node pairs in all operating states. For each $o \in O$ and $s, t \in V - o$ denote by $\pi^o_{s,t} \in \mathbb{R}^+$ the distance between s and t in operating state o with respect to w. It is well known from shortest path theory, that the metric inequalities

$$\pi^o_{s,u} + w_{(u,v)} - \pi^o_{s,v} \geq 0, \quad \forall\, o \in O,\, s \in V^o,\, (u,v) \in A^o$$

are satisfied and hold with equality if and only if (u, v) is on a shortest (s, v)-path in operating state o. Since scaling the weights does not change the shortest paths, we can find a scaling factor $\alpha > 0$ such that for weights αw the left-hand-side of all strict inequalities is at least 1.

Now suppose we are given integer variables p that define a path configuration, that is not necessarily admissible but satisfies all subpath constraints. Consider the following linear program, where w and π are variables (while p is fixed).[3]

$$min \sum_{(u,v)\in A} w_{(u,v)} \tag{22}$$

$$\pi^o_{s,u} + w_{(u,v)} - \pi^o_{s,v} = 0 \qquad \forall\, o \in O,\, s \in V^o,\, (u,v) \in A^o \text{ with } p^{s,v,o}_{(u,v)} = 1 \tag{23}$$

$$\pi^o_{s,u} + w_{(u,v)} - \pi^o_{s,v} \geq 1 \qquad \forall\, o \in O,\, s \in V^o,\, (u,v) \in A^o, \text{ with } p^{s,v,o}_{(u,v)} = 0 \tag{24}$$

$$w_a \geq 1 \qquad \forall\, a \in A \tag{25}$$

$$\pi^o_{s,v} \geq 0 \qquad \forall\, o \in O,\, s, v \in V^o,\, s \neq v \tag{26}$$

It follows directly that this linear program has a (primal feasible) solution if and only if the given variables p correspond to an admissible path configuration and, if this is the case, are unique shortest paths with respect to the weights w computed.

Otherwise, if the linear program has no solution, there are no routing weights inducing the given configuration of paths, i.e., the configuration is not admissible. In that case, the linear program contains an infeasible system of rows $I = I_1 \cup I_0$ consisting of some equalities $I_1 \subseteq \{(s, (u, v), o) \mid p^{s,v,o}_{(u,v)} = 1\}$ and inequalities $I_0 \subseteq \{(s, (u, v), o) \mid p^{s,v,o}_{(u,v)} = 0\}$. Since I is an infeasible system, for no admissible path configuration all $p^{s,v,o}_{(u,v)}$ with $(s, (u, v), o) \in I_1$ can have the value 1 and all $p^{s,v,o}_{(u,v)}$ with $(s, (u, v), o) \in I_0$ in I can have the value 0 simultaneously. Hence, the inequality

$$\sum_{(s,(u,v),o)\in I_1} p^{s,v,o}_{(u,v)} - \sum_{(s,(u,v),o)\in I_0} p^{s,v,o}_{(u,v)} \leq |I_1| - 1, \tag{27}$$

which is violated by the given non-admissible path configuration p, is valid for all admissible path configurations and can be added to the problem formulation.

Note that, in general, these infeasibility inequalities (27) are not facet defining. They cut off only one specific non-admissible sub-configuration of paths. To use them efficiently in a branch-and-cut framework, they have to be strengthened. Otherwise they will become active only too deep in the branching tree to be useful. In our implementation, we try to reduced their support as far as possible. First, we use only irreducible infeasibility systems of rows (set-inclusion minimal infeasibility systems), second, we apply some easy reductions that exploit the problem structure. For example, such an infeasibility systems usually contains both an entry $(s, (u, v), o) \in I_1$ and one or more entries $(s, (u', v), o) \in I_0$. Since, if $p^{s,v,o}_{(u,v)} = 1$, the variables $p^{s,v,o}_{(u',v)}$ are already forced to 0 by the flow balance constraints (14), they need not be included in the infeasibility system's inequality (27).

[3] In practice, integer routing weights between 1 and $2^{16} - 1$ are required. It is easy to see that fractional weights w can always be scaled by a big enough value and then rounded to integer weights w' that induce the same shortest paths (see also [BAG00]). In order to decide whether a path configuration is admissible it thus suffices to consider fractional routing weights. To provide integer routing weights for the final solution, we solve the linear system with integer w-variables and an artificial objective function minimizing the total weight once in the post-processing of each solution. In practice, these integer programs are very easy to solve.

If the OSPF routing must be symmetric, like in the G-WiN, one usually wants to provide equal routing weights for both directions of an edge. This is achieved by adding the equalities

$$w_{(u,v)} = w_{(v,u)} \quad uv \in E \tag{28}$$

to the linear system (23)–(26).

4.4 Utilization minimization

The variables and constraints presented so far are sufficient to model the cost minimization variants of the backbone network design problem as a mixed-integer linear program. For the utilization minimization variant, we need some further variables and constraints.

For each load group $j \in J$, we introduce a continuous variable $\lambda^j \in \mathbb{R}$, $0 \leq \lambda^j \leq \overline{\lambda^j}$, for the maximum utilization attained by a component c on some edge e in operating state o with $(c, e, o) \in j$. The upper bound $\overline{\lambda^j}$ for that variable is used to specify initially a maximum feasible utilization. The objective is

$$\min \sum_{j \in J} \alpha^j \lambda^j, \tag{29}$$

where α^j is the non-negative objective coefficient associated with load group j.

For each $j \in J$, $e \in E$, and $c \in C$ with $(c, e, o) \in j$ for some $o \in O$, we introduce variables for the maximum usable routing capacity $y_e^{c,j} \geq 0$. The value of these variables is the amount of flow that can be routed through the component in this operating state without increasing the current maximum utilization for the load group.

The maximum usable capacities cannot be larger than the original components capacity, if installed, times the maximum attained utilization of the load group. This can be expressed by the following two variable-upper-bound inequalities:

$$y_e^{c,j} \leq \lambda^j k_e^{c,cap} \qquad \forall\, j \in J, c \in C, e \in E \text{ with } (c, e, o) \in j \text{ for some } o \in O \tag{30}$$

$$y_e^{c,j} \leq z_{uv}^c (\overline{\lambda^j} k_e^{c,cap}) \quad \forall\, j \in J, c \in C, e \in E \text{ with } (c, e, o) \in j \text{ for some } o \in O \tag{31}$$

In the capacity constraints (15) the original capacities provided by the installed edge components are replaced by the maximum usable capacities. This yields for utilization minimization the new capacity constraints

$$\sum_{s,t \in V^o} d^{s,t} \cdot p_{(u,v)}^{s,t,o} \leq \sum_{c \in C : j(c,uv,o) \neq \emptyset} y_{uv}^{c,j(c,uv,o)} \quad \forall\, o \in O, (u, v) \in A^o, \tag{32}$$

where $j(c, uv, o)$ denotes the load group that (c, uv, o) belongs to or $\emptyset$, if it belongs to no load group.

5 Computation

In this section we report on the methods used to solve the models shown in the previous sections. Also, we explain why these algorithms where choosen.

5.1 Access network

In all cases we studied, the access network planing problem was highly restricted. This is not surprising, because usually the locations that can be choosen as backbone or intermediate nodes are limited due to

technical, administrative and political reasons. The biggest obstancle always was the unavailability of sensible cost data for the potential connections. Also, in all projects we encountered some peculiar side constraints that were specific to the project. Therefore, we needed a very flexible approach and used a general IP-Solver. And, at least for the data used in the projects, it was possible with some preprocessing to solve the access network planing problems with CPLEX [CPL01], SIP [Mar98], or a comparable MIP-Solver.

Unfortunately, the solution times heavily depend on the actual data and objective function, especially on the relation between connection costs and assembly stage costs and on how tight the resource capacities are. In the G-WiN all locations were already built and the number of backbone and intermediate nodes to choose was fixed as a design criteria. So only the connetion costs had to be regarded in the objective function. In one of the other projects this situation reversed: the transportation network was already installed and only the cost for the equippment at the nodes had to be considered. In Section 6 we show some examples on how difficult the problems can get, if we allow all connections between the nodes and have several tight resources.

5.2 Backbone network

In order to solve the backbone network planning problem a special branch-and-cut algorithm based on the MIP formulation presented in Section 4 was developed and implemented in C++. CPLEX [CPL01] or SoPlex [Wun96] can be used to solve the linear programming relaxations, the branch-and-cut tree is managed by our software.

The initial LP relaxation contains all component variables and all resource constraints (10), (11), (12). If the objective is utilization minimization, all load group variables and all usable capacity variables are in the initial LP relaxation as well as the associated variable-upper-bound constraints (30) and (31).

If the binary arc–flow variables of all demands and all operating states are considered, the LP relaxation becomes too large to be used. We only considerer the arc–flow variables for all demands in the normal operating state and for the biggest demands in the node failure states, i.e., only for demands with a value of least 50% of the biggest demand value. Arc–flow variables for edge failure states are not in the formulation. Most edge failure states are dominated by some node failure states, already. Numerous experiments with real world data revealed that using only this restricted variable set is a good tradeoff between the quality of the relaxation and the time necessary to obtain good solutions. The flow balance constraints (14) and, the edge capacity constraints (15) or (32) are in the initial formulation, if at least one arc–flow variable in their support is. The routing symmetry constraints (16), of course, are not explicitly generated in practice. Instead, arc–flow variables are generated only for one of the directions s to t or t to s and then used to model the paths for both directions. The path length inequalities (17), if necessary, are also in the initial LP. Clearly, for all arcs that do not belong to any short (s,t)-path or those starting in t or ending in s, the corresponding arc–flow variables must be 0 and are not included the model.

Although there is only a polynomial number of subpath constraints (18), (19), and (20) and operating state coupling constraints (21), we generate these inequalities only if violated to keep the size of the relaxation as small as possible. Also, we only use coupling inequalities (21) that link the normal operating state to some failure state. Separating coupling inequalities between two failure states is too time consuming and many of these inequalities are already implied by the coupling constraints between the normal operating state and the two failure states.

At each node of the branch-and-cut tree we iteratively separate violated inequalities and resolve the LP relaxation for a limited number of times (≤ 5), as long as there is substantial increase ($\geq 1\%$) in the optimium LP-value. In each such iteration we separate the following inequalities in the order given below, until at most 50 inequalities are separated:

1. subpath inequalities (18), (19), and (20) for the normal operating state,

2. induced cover inequalities [Boy93] for the knapsacks given by the edge capacity constraints (15) or (32) in the normal operating state [4],

3. cover inequalities [BZ78, NV94] for the knapsacks given by the resource constraints (10), (11), and (12),

4. subpath inequalities (18), (19), and (20) for failure operating states,

5. induced cover inequalities for the knapsacks given by the edge capacity constraints (15) or (32) for failure operating states,

6. operating state coupling inequalities (21) between the normal and some failure operating state, and

7. IIS inequalities (27).

This strategy separates first those inequalities that are computationally easier to separate and give the best improvement in the LP bound. It proved to be well suited in our experiments.

If, at some branch-and-cut node, the final LP solution is fractional, our basic strategy is to branch on a "good" arc–flow variable of a big demand. From those demands, whose value is no less than a certain percentage (0.9) of the biggest demand with fractional arc–flow variables, we choose the arc–flow variable whose fractional value is closest to a target value (0.8). Flow variables for the normal operating state are preferred: When choosing the next branching variable we divide all demands by a factor of 5 for failure state flow variables. At every branch-and-cut node with depth $3k$, $k \in \mathbb{N}$, we branch on a fractional component variable, if there is one. In such a case, global components are always preferred to edge and these to node components, and then the fractional variable whose component provides the biggest routing capacity is chosen.

The next node to explore in the branch-and-cut tree is selected by a mix of best-dual-bound and dive strategy. For a given number of iterations (32) we choose the node with the best dual bound. Thereafter, we dive for a good feasible solution in the tree, starting at a best-dual-bound node and then always chosing the child node whose arc–flow variable was set to 1 or edge component variable was rounded up in the last branching. We do not backtrack on the dive path. If a feasible solution was found or the last child node is infeasible, we switch back to the best-dual-bound strategy and continue.

Two primal heuristics are used at the branch-and-cut nodes. They are applied only at nodes with depth 2^k, $k \in \mathbb{N}$, reflecting the fact that the "important" branches, that fix big demands' flow variables or large edge components' variables, are performed first, while deeper in the branch-and-cut tree only small demands' flow variables are fixed. Both heuristic first generate initial routing weights and compute a shortest path routing with respect to these weights. Then, a (mixed-) integer programming solver is used to compute a hardware configuration minimizing the original cost or utilization objective function plus the total violation of edge capacity constraints by this routing. These integer programs are fairly easy to solve in practice, they only contain the component variables and ressource constraints. In a last step, the initially routing weights are adopted to the topology computed by the integer program and perturbed to make all shortest paths unique.

Our first heuristic takes a linear combination of the dual variables of the edge capacity constraints (15) or (32) over the different operating states as initial routing weights. The second heuristic utilizes the linear program (22)–(26) to compute initial routing weights. In contrast to the IIS inequality separator, the heuristic initializes the metric (in-)equalities (23) and (24) only for those arc–flow variables $p^{s,v,o}_{(u,v)}$ that are integer or near-integer (≤ 0.1 or ≥ 0.9) in the current fractional solution. All other metric inequalities (23) or (24) are "deactivated" by setting their sense and right hand side to "≤ 0". If this LP has a feasible solution, the computed routing weights induce at least all the near-integer routing paths at the current branch-and-cut node. If it has no solution, we generate an IIS inequality (27) cutting off a non-admissible sub-configuration of the near-integer paths. The heuristic works as a separator, too.

[4] The precedence constraints in these knapsacks are the subpath constraints among the paths using the edge.

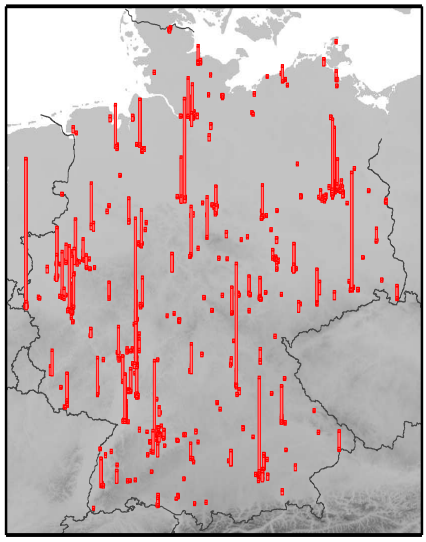

Figure 4: G-WiN-1 sites with aggregated traffic demands

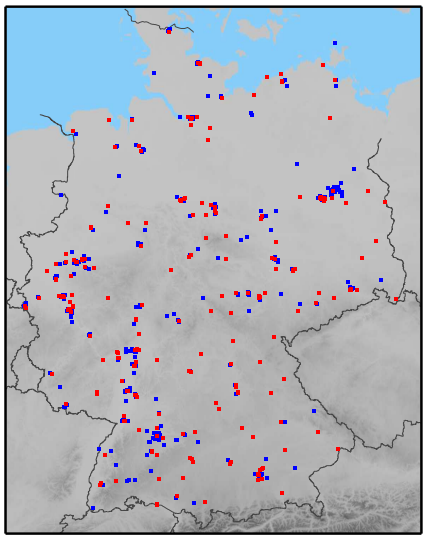

Figure 5: G-WiN-1 sites, potential level-1/2 sites red

If the objective is to minimization the average utilization, we apply additional bound strengthening techniques to speed up the computation. For each load group j, we store an upper bound $\overline{\lambda^j}^*$ on the corresponding maximum utilization of the *optimal* solution and update these bounds during the optimization process. Initially, $\overline{\lambda^j}^* = \overline{\lambda^j}$. Whenever a new feasible solution with average utilization λ^* is found, we tigthen these bounds by setting $\overline{\lambda^j}^* := min\{\overline{\lambda^j}^*, \lambda^*/\alpha^j\}$ and we update the variable upper bound constraints (31) for these new bounds $\overline{\lambda^j}^*$. Note that this operation may exclude feasible but non-optimal solutions from the remaining solution space; but we are interested only in the optimal solution and not in the entire feasible solution space. Indirectly, this bound strengthening also tightens the capacity constraints (32). Thus, in the following separation attempts we may find much stronger cover inequalities for the associated knapsacks than we would find without the bound strengthening. But, again, these inequalities are valid only for the optimal solution, not for all feasible solutions of the original problem.

6 Results

The application of our main interest was the planning of the german research network G-WiN. The G-WiN startet in 2000 with about 759 demand locations, see Figure 4, 261 of them could be used as intermediate or backbone nodes, see Figure 5.

It was decided from the administration, that there should be about 10 backbone nodes and two intermediate node per backbone node. Several sets of demand data were generated. For the initial access and backbone network planning accounting data from the predecessor network B-WiN was used, later, for the operational replanning of the G-WiN backbone network, accounting data in the latest G-WiN network were available. These traffic mesurements were scaled to anticipate future demands.

In the access network planning step, several scenarios were optimized and assessed according to the stability of the solutions for increasing demands. The backbone nodes were then successively selected. For this reason we can not provide running times for the original problems. We generated some data sets with the original data and varying costs and resources. As can be seen in Table 1, instances of identical size can vary strongly in running time (CPLEX 8.0 MIP solver on an 2.5GHz Pentium4, Linux). The harder instances were always those with scarcer resources, a higher number of assembly stages, or a more difficult

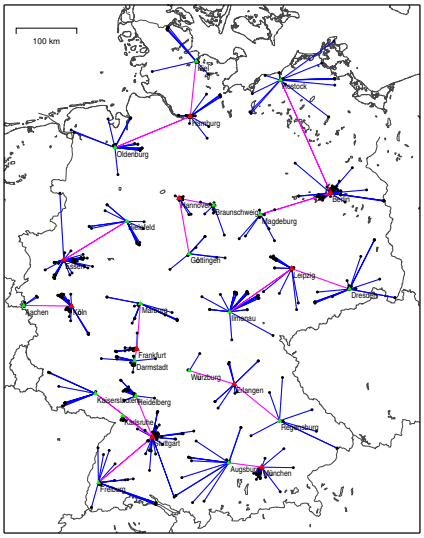

Figure 6: G-WiN-1 access network solution

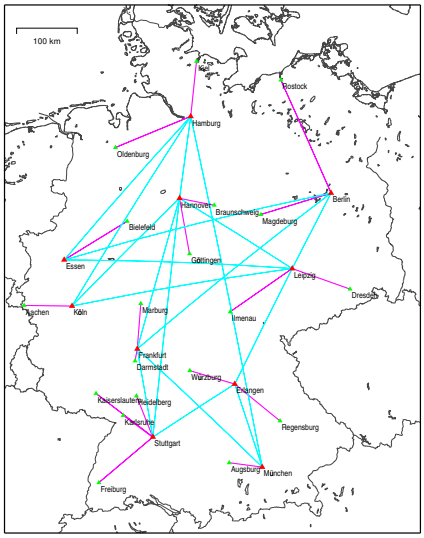

Figure 7: G-WiN-1 core backbone and level-1–level-2 links

Name	Vars	Cons	Nonz	B&B	Time (hh:mm:ss)
DFN1	6062	771	12100	235	6
DFN2	5892	747	11781	13670	27:05
DFN3	6062	771	12100	>40000	>3:00:00
TV1	1818	735	5254	0	<1
TV2	1818	735	5254	24	<1
SL1	7188	3260	27280	267	13
SL2	7188	3260	27280	7941	1:26
KG1	8230	5496	42724	511	2:56
KG2	8230	5496	42724	3359	3:25
BWN1	13063	9920	57688	372	1:48
BWN2	13063	9920	57688	18600	5:54

Table 1: Results for the access network planning problem

cost structure. The solution for the G-WiN access network planning problem is shown in Figure 6.

It should be noted that access link cost functions that do not depend on the distance can lead to very strange-looking optimal solutions. We encountered this, when we had costs that were constant for a certain range. These solutions may be difficult to verify and crossing access network link may be very hard to explain to a practitioner.

For the G-WiN backbone planning, the underlying network topology was a complete graph on 10 nodes, corresponding to the virtual private STM backbone network inside Germany, plus one additional node linked to two of these ten nodes, corresponding to the 'uplink' to other networks via two gateways. The capacities on the two gateway links and the 'configuration' of the uplink-node were fixed. On each of the other links, one capacity configuration from a given set of configurations could be installed. Depending on the specific problem instance, these sets represented various subsets of the STM-hierarchy. Several types of IP router cards and SDH cards had to be considered as components at the nodes. In order to install a certain capacity configuration on an edge, appropriate IP router and SDH interfaces must be provided by the router and interface cards at the two terminal nodes. Additional IP-interfaces had to be provided

Problem	LP-Opt	best LB	best sol	Time (hh:mm:ss)	B&B
G-WiN-2 nos	0.482	0.576	0.576	38.0	775
G-WiN-3 nos	0.054	0.106	0.106	7.9	48
G-WiN-4 nos	0.030	0.057	0.057	18.4	50
Atlanta nos	0.633	0.834	0.834	6.8	36
EP98 nos	0.183	0.235	0.235	1:29.4	204
NSF nos	0.685	0.686	0.686	23:48.3	3453
G-WiN-2 fail	2.610	3.059	3.059	14:17.1	1350
G-WiN-3 fail	0.104	0.273	0.464	2:00:00	140
G-WiN-4 fail	0.093	0.137	0.142	2:00:00	1284
Atlanta fail	1.265	1.669	1.669	13.8	70
EP98 fail	0.778	0.962	1.271	2:00:00	1406
NSF fail	0.731	0.745	0.746	2:00:00	2078

Table 2: Results for backbone utilization minimization problems

at the nodes in order to connect the backbone level to the access level of the network. Of course, every node could hold only a limited number of router and interfaces cards. Besides these local restrictions, for both, edge capacities components as well as node interface and router card components, there were also several global restrictions. For example, the totally installed edge capacity and the number of installations of each edge capacity type and each card type were bounded. Besides these rather natural constraints, there were also more complicated ones limiting the number of certain reconfigurations for the operational network replanning. These constraints were modeled via artificial global and local resources that account for changes in the number of installed hardware components. Finally, the routing path length for each traffic demand was bounded by 2 or 3 hops in the normal operating state.

For the design of the initial G-WiN, the objective was to minimize the total cost of the node and edge hardware components. It sould be noted that the costs associated with the edge capacity types and IP router or SDH interface cards did not depend on the specific edge or node where these components were installed. For the operational replanning of the G-WiN backbone, the objective was to minimize an average of the maximum utilizations for the following four load groups: all edge components on national edges in the normal operating state, all edge components on the two gateway edges in the normal operating state, all edge components on national edges in the failure operating states, and all edge components on the two gateway edges in the failure operating states.

The cost optimal solution for the G-WiN-1 backbone design problem, together with the backbone-intermediate links, is shown in Figure 7.

Typical computational results for utilization minimization and cost minimization problems are presented in Tables 2 and 3, respectively. The suffix *fail* or *nos* in the names indicate whether or not failure operating states are considered. The computations were performed on an 1.7GHz Pentium4, running the Linux operating system and CPLEX 7.5 as LP solver, with a total time limit of two hours. The column LP-Opt displays the values of the initial linear programming relaxations, best LB and best sol show the best lower bounds and the best solutions after the branch-and-cut algorithm. Iime and B&B report the time and the number of evaluated branch-and-cut nodes until the optimal solution was found or the time limit was reached.

The first group of instances are real world planning problems from the German research network G-WiN, as described above. The other group of problem instances, i.e., Atlanta, EP98, and NSF, are based on real network topologies with 15 nodes and 22 edges, 13 nodes and 24 edges, and 14 nodes and 21 edges, respectively, and real world traffic data. To these data sets we added artificial capacity structures similar to the STM-hierarchy, but scaled according to the demand values in the data sets. There are no node or global components and no node or global restrictions in these instances.

Problem	LP-Opt	best LB	best sol	Time (hh:mm:ss)	B&B
G-WiN-2 nos	72.41	76.00	76.00	8	21
G-WiN-3 nos	240.00	240.00	240.00	6	23
G-WiN-4 nos	272.00	272.00	272.00	3	3
Atlanta nos	56.44	104.39	107.00	2:00:00	3073
EP98 nos	165.05	176.35	184.17	2:00:00	2114
NSF nos	229.48	324.42	475.00	2:00:00	198
G-WiN-2 fail	72.41	76.00	76.00	3:07	19
G-WiN-3 fail	320.00	320.00	320.00	24:12	278
G-WiN-4 fail	272.00	272.00	272.00	56	5
Atlanta fail	65.42	126.46	175	2:00:00	960
EP98 fail	179.54	202.25	309.23	2:00:00	623
NSF fail	318.60	389.32	735.00	2:00:00	115

Table 3: Results for backbone cost minimization problems

Looking at the results for the utilization minimization in Table 2, we see that the smaller instances, especially those where no failure operating states were considered, could be solved to optimality within resonable times. For bigger instances, which also consider failure operating states, the algorithm in general finds very good solutions (which could not be improved by other heristic methods) within the given time bound, although it sometimes fails to prove a reasonable lower bound. We want to point out that the best solutions shown in the tables were identified very quickly, which was one of our main goals in the design of the primal heuristics and the branching strategy. Typically, the algorithm improves the dual bound pretty fast during the first iterations, when branches are performed on the biggest demands and edge components. Later, after a few hundered iterations, there is almost no further improvement in the dual bound. The same general behavior was observed for cost minimization problems.

Comparing the results for utilization minimization and cost minimization, the G-WiN instances for cost minimization are much easier to solve than their utilization counterparts. The reason is that in these instances the traffic demands are relatively small compared to the potential edge capacities and that the global reconfiguration restrictions limit the number of feasible topologies. Thus, finding a good routing is not so important in the cost minimization variants of these instances. For the other problem instances, it seems that the utilization minimization variants are easier to solve than their respective cost minimization counterparts. This is a result of the bound strengthening described in the previous section. This technique, which can be applied only for utilization minimization, leads to a substantial reduction of the remaining search space whenever a new solution is found, and thus speeds up both the finding of the optimal solution and the verification of its optimality.

Acknowledgements

We would like to thank DFN and BMBF for funding this project and our colleagues at DFN, in particular M. Pattloch, for fruitful discussions and excellent cooperation.

References

[AGW97] D. Alevras, M. Grötschel, and R. Wessäly, *Capacity and survivability models for telecommunications networks*, Tech. Report SC 97-24, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1997.

[ALLQ96] K. Aardal, M. Labbé, J. Leung, and M. Queyranne, *On the two-level uncapacitated facility location problem*, INFORMS Journal on Computing **8** (1996), 289–301.

[BAG00] W. Ben-Ameur and E. Gourdin, *Internet routing and related topology issues*, submitted to SIAM J. Discrete Mathematics, 2000.

[BAGL00] W. Ben-Ameur, E. Gourdin, and B. Liau, *Internet routing and topology problems*, Proceedings of DRCN2000 (Munich), 2000.

[BCGT98] D. Bienstock, S. Chopra, O. Günlük, and C-Y. Tsai, *Minimum cost capacity installation for multicommodity network flows*, Mathematical Programming **81** (1998), 177 – 199.

[BGW98] A. Bley, M. Grötschel, and R. Wessäly, *Design of broadband virtual private networks: Model and heuristics for the B-WiN*, Robust Communication Networks: Interconnection and Survivability (N. Dean, D. F. Hsu, and R. Ravi, eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 53, AMS, 1998, pp. 1–16.

[BK00] A. Bley and T. Koch, *Optimierung des G-WiN*, DFN-Mitteilungen (2000), no. 54, 13–15.

[BKSTG00] L. Berry, S. Köhler, D. Staehle, and P. Tran-Gia, *Fast heuristics for optimal routing in large IP networks*, Tech. report, Department of Computer Science, University of Würzburg, 2000.

[Boy93] E.A. Boyd, *Polyhedral results for the precedence-constrained knapsack problem*, Discrete Applied Mathematics (1993), no. 41, 185 – 201.

[BZ78] E. Balas and E. Zemel, *Facets of the knapsack polytope from minimal covers*, SIAM Journal on Applied Mathematics **34** (1978), 119 – 148.

[CPL01] ILOG CPLEX Division, 889 Alder Avenue, Suite 200, Incline Village, NV 89451, USA, *ILOG CPLEX 7.5 reference manual*, 2001, Information available at http://www.cplex.com.

[CW92] J. Crowcroft and Z. Wang, *Analysis of shortest-path routing algorithms in a dynamic network environment*, ACM SIGCOM Computer Communication Review **22** (1992), no. 2, 63–71.

[ERP01] M. Ericsson, M.G.C. Resende, and P.M. Pardalos, *A genetic algorithm for the weight setting problem in OSPF routing*, Tech. report, AT&T Labs Research, 2001.

[FGL+00] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, *NetScope: Traffic engineering for IP networks*, IEEE Network **14** (2000), no. 2, 11 – 19.

[FMdS+96] C.E. Ferreira, A. Martin, C.C. de Souza, R. Weismantel, and L.A. Wolsey, *Formulations and valid inequalities of the node capacitated graph partitioning problem*, Mathematical Programming **74A** (1996), 247–266.

[FT00] B. Fortz and M. Thorup, *Internet traffic engineering by optimizing OSPF weights*, Proceedings of IEEE INFOCOM 2000, 2000.

[GMS95] M. Grötschel, C.L. Monma, and M. Stoer, *Design of survivable networks*, Handbooks in Operations Research and Management Science, vol. Network Models, ch. 10, pp. 617–672, North-Holland, 1995.

[Gou01] E. Gourdin, *Optimizing internet networks*, OR/MS Today (2001), 46–49.

[Hal96] L. Hall, *Experience with a cutting plane algorithm for the capacitated spanning tree problem*, INFORMS Journal on Computing **8** (1996), 219–234.

[HD01] K. Holmberg and D.Yuan, *Optimization of internet protocol network design and routing*, Tech. report, Linköping University, 2001.

[KM98] T. Koch and A. Martin, *Solving Steiner tree problems in graphs to optimality*, Networks **32** (1998), 207–232.

[LW93] F.Y.S. Lin and J.L. Wang, *Minimax open shortest path first routing algorithms in networks suporting the SMDS service*, Tech. report, Bell Communications Research, 1993.

[Mar98] A. Martin, *Integer programs with block structure*, Habilitations-Schrift, Technische Universität Berlin, 1998.

[MD01] S. Melkote and M.S. Daskin, *Capacitated facility location/network design problems*, European Journal of Operations Research **129** (2001), 481–495.

[MF90] P. Mirchandani and R. Francis (eds.), *Discrete location theory*, Wiley, New York, 1990.

[NV94] G. L. Nemhauser and P. H. Vance, *Lifted cover facets of the* $0-1$ *knapsack polytope with GUB constraints*, Operations Research Letters **16** (1994), 255 – 263.

[PLPL00] K Park, K. Lee, S. Park, and H. Lee, *Telecommunication node clustering with node compatibility and network survivability requirements*, Management Science **46** (2000), 263–374.

[Pry02] M. Prytz, *On optimization in design of telecommunications networks with multicast and unicast traffic*, Ph.D. thesis, Royal Institute of Technology, Stockholm, Sweden, 2002.

[SKK00] D. Staehle, S. Köhler, and U. Kohlhaas, *Towards an optimization of the routing parameters for ip networks*, Tech. report, Department of Computer Science, University of Würzburg, 2000.

[Wun96] R. Wunderling, *Paralleler und objektorientierter simplex*, Tech. Report TR 96-09, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1996, Information available at http://www.zib.de/Optimization/Software/Soplex.