Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7, D-14195 Berlin

A Network Dimensioning Tool

Dimitris Alevras Martin Grötschel Roland Wessäly

Preprint SC 96–49 (December 1996)

A Network Dimensioning Tool

D. Alevras M. Grötschel R. Wessäly

December 12, 1996

Abstract

Designing low cost networks that survive certain failure situations belongs to one of the prime tasks in the telecommunications industry. In this paper we describe a mathematical model integrating several aspects of survivability that are elsewhere treated in a hierarchical fashion. We present mathematical investigations of this model, a cutting plane algorithm, as well as several heuristics for its solution. Moreover, we report computational results with real-world data.

The problem we address is the following. Suppose, between each pair of nodes in a region, a communication demand is given. We want to determine the topology of a telecommunication network connecting the given nodes and to dimension all potential physical links. For each link, the possible capacities are restricted to a given finite set. The capacities must be chosen such that the communication demands are satisfied, even if certain network components fail, and such that the network building costs are as small as possible. Moreover, for each pair of nodes and each failure situation, we want to determine the paths on which the demand between the nodes is routed.

Keywords: Telecommunication Network Design, Survivable Networks, Network Capacity Planning, Cutting Plane Algorithm, Heuristics, Routing

Mathematical Subject Classification (1991): 90B12, 90C11, 90C27, 90C90, 94A99

1 Introduction

Due to deregulation, telecommunication has become a highly competitive area: Low cost and high quality of its service are vital for the success of a company. For a mobile telecommunications provider, such as our project partner `e-plus` Mobilfunk GmbH, good service includes high connection quality (low interference and background noise) and high network survivability (low impact of component failures). Both parameters depend on the quality of the technical equipment used, but they also depend heavily on a proper planning of the network, an issue we address in our paper. We focus in this paper on the problem of designing and dimensioning a network (the `e-plus` backbone network) in such a way that it survives certain component failures, has low cost and is easy to manage.

The process of dimensioning a network is based on the application of methods from such diverse areas as statistics, economics, electrical engineering and operations research. We contribute to the last aspect and show how models and solution techniques from integer programming can help designing good quality telecommunication networks.

Figure 1 indicates architecture of a typical mobile-telecommunications network. The architecture consists of two layers: the switching layer and the transport layer.

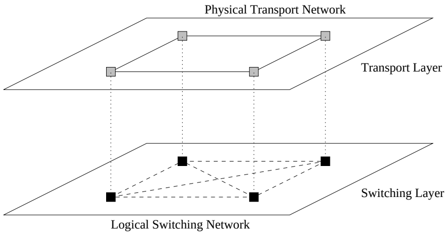

Figure 1: Typical mobile-telecommunications network architecture

The network planning process starts with a forecast of point-to-point demands in terms of Erlang. Based on Grade-of-Service requirements, such as the number of blocked calls and routing rules, the switching layer planning (whose details we do not describe here) results in a *logical transport network*. Each link in the transport network expresses the telecommunication demands between its end nodes in terms of numbers of channels; in our case one channel has a capacity of 64 kbit/s.

The transport layer planning consists of deciding the capacities of the physical links (taking advantage of multiplexing capabilities), and of providing the routing tables that map the logical demands of the transport network to paths in the dimensioned physical network. If survivability is required at this layer, then the routing tables also include routings for failure situations.

There are several ways to implement survivability in the physical network. One method is to consider the **uncapacitated network design problem**. This deals with connectivity requirements only and is treated, for instance, in Monma and Shallcross [MS89], Monma, Munson and Pulleyblank [MMP90], Grötschel, Monma and Stoer [GMS92a, GMS92b], and Stoer [Sto92]. In the **capacitated network design problem** the demands between pairs of nodes must be taken into account, in addition to the connectivity of the network.

In this paper, we study the problem of selecting from a **discrete set of possible capacities** which one to install on each link of the physical network so that each demand can be routed (even in the case of a single node or single edge failure) and the capacity installation cost is minimum. Additional restrictions to the percentage of a demand routed through a particular node or edge of the network and to the length of the paths between two demand nodes are considered. We model this problem as a mixed-integer linear program and present a cutting plane algorithm for its solution. Due to the difficulty of the problem, finding optimal solutions is out of reach. But, by combining LP-relaxations and heuristics, we obtain low cost solutions with a quality guarantee, i.e., a bound on the gap between the solution value and the (unknown) optimal one.

Variants of our problem have been considered in the literature. Minoux [Min81] considers survivability in a generalized multicommodity-flow model, but the allowed capacities are non-discrete. Instances up to 20 nodes are solved within an accuracy of 5–10%. In cooperation with *France Telecom* Lisser, Sarkissian and Vial [LSV95] developed another model including non-discrete capacities and survivability. In [LSV95] two survivability models are presented, both different from ours. In both models, part of the demand is routed in case of a failure in a separate network, called spare network. In case of a network failure (node or edge failure) the *local-survivability* model routes only the failing flow, and the *global-survivability* model routes only the affected demands. Tests with up to 53 nodes, 79 edges and 1378 demands are reported. Several models in the literature consider the installation of discrete capacities without addressing survivability issues, see e.g. Bienstock and Günlük [BG95], and Magnanti, Mirchandani, and Vachani [MMV95]. These latter studies, however, restrict the possible capacities to multiples of two base-capacities. Dahl and Stoer [DS92, DS94] studied a problem for *Norwegian Telecom Research* that is similar to ours but without imposing length restrictions on the paths between demand nodes.

The model we present in this paper was developed in cooperation with `e-plus` Mobilfunk GmbH, one of the three mobile-telecommunication service providers in Germany. The solution methodology we describe later forms the core of a network dimensioning tool, called DISCNET (**Di**mensioning of **S**urvivable **C**ellular-Phone **Net**works), that we implemented and that is in use at `e-plus` for its transport network planning.

The remaining part of the paper is organized as follows. In the next section we formally define the problem and present the model. A high-level description of the solution approach is given in Section 3, while in Section 4, we describe the related polytopes and classes of valid inequalities for these polytopes. A number of subproblems that come up are multicommodity-flow problems. Details of the multicommodity-flow algorithms are presented in Section 5. Section 6 contains a brief description of the heuristic approaches. In Section 7 we present computational results and in Section 8 we discuss three methods of realizing survivability in the network. We conclude with some open questions in Section 9.

2 The Model

The problem we consider can be described as follows. The input consists of two graphs on the same node-set V, the **supply graph** $G = (V, E)$ and the **demand graph** $H = (V, D)$. The set V consists of the nodes of the logical transport network. In our application, V is the set of MSC locations; in some cases BSC locations are included (MSC $\widehat{=}$ Mobile Switching Center, BSC $\widehat{=}$ Base Station Controller). The edge-set E of the supply graph is the set of all physical links that may potentially be used. Different types of links (e.g., microwave, leased lines etc.) are represented by **parallel edges**. The demand graph contains an edge whenever there is a positive demand between its two end nodes.

In our practical application, it was natural to assume that every edge $e \in E$ of the supply graph is already equipped with an initial capacity $M_e^0 \in \mathbb{Z}_+$ (possibly $M_e^0 = 0$) of cost $K_e^0 = 0$, the so-called **free capacity**. For each $e \in E$, the following further data are given:

- $T_e \in \mathbb{Z}_+$, where T_e is the **number of possible additional capacities** that can be installed,
- $M_e^t \in \mathbb{Z}_+$, $1 \leq t \leq T_e$, the **potential capacities**, (we assume that $M_e^0 < M_e^1 < \cdots < M_e^{T_e}$),
- $K_e^t \in \mathbb{Q}_+$, $1 \leq t \leq T_e$, the **cost** of installing capacity M_e^t .

It has turned out to be useful to call the capacities $M_e^1, \ldots, M_e^{T_e}$ **breakpoint capacities**, and hence T_e the number of **breakpoints**, and to consider the **incremental capacities** and **costs**

- $m_e^t := M_e^t - M_e^{t-1}$, $1 \leq t \leq T_e$,
- $k_e^t := K_e^t - K_e^{t-1}$, $1 \leq t \leq T_e$,

instead of the original values. For notational reasons, we set $m_e^0 := M_e^0$ and $k_e^0 := K_e^0$.

For each edge $uv \in D$ of the demand graph, the value

- $d_{uv} \in \mathbb{Z}_+$ is the **communication demand** between nodes u and v.

Moreover, the network designer specifies, for each $uv \in D$, the following parameters:

- δ_{uv}, $0 < \delta_{uv} \leq 1$, the **diversification** parameter; δ_{uv} is the maximum fraction of the demand d_{uv} allowed to flow through any supply edge or node (other than nodes u and v),
- ρ_{uv}, $0 \leq \rho_{uv} \leq 1$, the **reservation** parameter; ρ_{uv} is the fraction of the demand d_{uv} that must be satisfied in a single node or a single supply edge failure,
- $\ell_{uv} \in \mathbb{Z}_+$, the **path length restriction**; ℓ_{uv} is the maximum number of supply edges allowed in any path on which demand between u and v is routed.

For the network we want to design we also wish to determine the routings of the demands for each **operating state** s of the network. The operating states are

- the **normal** state ($s = 0$), which is the state with all nodes and all edges operational, and
- the **failure** states, which are the states with a single node u ($s = u$) or a single edge e ($s = e$) nonoperational.

We denote by $G_s = (V_s, E_s)$ the supply graph for the operating state s, where V_s is the set of nodes that are still operational in operating state s, and, likewise, E_s is the set of the operational edges in operating state s. Similar notational conventions apply to the demand graph.

Our goal is to choose, for each supply edge, a capacity such that there exist routings that satisfy all the restrictions and such that the sum of all capacity installation costs is as small as possible.

We model our network dimensioning problem as a **mixed-integer linear programming problem** with two types of variables, integer decision variables x_e^t and continuous routing variables $f(s, uv, P)$.

For each edge $e \in E$ we introduce an ordered set of **0/1 variables** $x_e^0 \geq x_e^1 \geq \cdots \geq x_e^{T_e}$. Since we assume that a free capacity M_e^0 is always installed, we set $x_e^0 := 1$. Choosing capacity M_e^τ , $0 \leq \tau \leq T_e$, is equivalent to setting $x_e^0 = x_e^1 = \cdots = x_e^\tau = 1$ and $x_e^{\tau+1} = \cdots = x_e^{T_e} = 0$.

For each operating state s and each demand edge $uv \in D_s$, let $\mathcal{P}(s, uv)$ denote the set of **valid** uv**-paths** in G_s. If s is the normal operating state, a uv-path in $G = G_0$ is valid if its length (number of edges) is at most ℓ_{uv}. We call such a path **short**. If s is a failure state then any uv-path in G_s is valid. For each operating state s, each edge $uv \in D_s$, and each path $P \in \mathcal{P}(s, uv)$, we define a variable $f(s, uv, P)$, called **flow** or **routing variable**, that represents the communication traffic between the nodes u and v routed on path P in operating state s.

We use the symbol x to denote the vector (in the Euclidean space of dimension $\sum_{e\in E}(|T_e|+1)$) whose components are the variables x_e^t introduced above; similarly f denotes the vector(in the Euclidean space of dimension $\sum_s \sum_{uv\in D_s} |\mathcal{P}(s, uv)|$) whose components are the routing variables $f(s, uv, P)$.

The objective is to minimize the total cost of installing the necessary capacities on the edges of the supply graph. This is formulated as

$$\min \sum_{e\in E} \sum_{t=1}^{T_e} k_e^t x_e^t \ . \tag{1}$$

To represent feasible choices of capacities and feasible routings, the vectors x and f must satisfy the following constraints. The 0/1-variables associated with a supply edge must satisfy the **ordering constraints**

$$1 = x_e^0 \geq x_e^1 \geq \cdots \geq x_e^{T_e} \geq 0 \qquad \text{for all } e \in E \ , \tag{2}$$

and the **integrality constraints**

$$x_e^t \in \{0, 1\} \qquad \text{for all } e \in E \text{ and } t = 1, \ldots, T_e \ , \tag{3}$$

by definition. For notational convenience, we introduce auxiliary variables

$$y_e := \sum_{t=0}^{T_e} m_e^t x_e^t \qquad \text{for all } e \in E \ , \tag{4}$$

representing the capacities installed on the supply edges.

For each operating state s and for each supply edge $e \in E_s$, the flow through e may not exceed its capacity. This trivial observation yields the **capacity constraints**

$$y_e \geq \sum_{uv\in D_s} \sum_{P\in\mathcal{P}(s,uv):e\in P} f(s, uv, P) \qquad \text{for all } s \text{ and all } e \in E_s \ . \tag{5}$$

The routing variables must be chosen in such a way that all the demands d_{uv} in the normal operating are satisfied. In any other operating state we require that the "reduced demands" $\rho_{uv} d_{uv}$ are met. Thus, the following **demand constraints**

$$\sum_{P \in \mathcal{P}(0,uv)} f(0, uv, P) = d_{uv} \qquad \text{for all } uv \in D \;, \tag{6}$$

$$\sum_{P \in \mathcal{P}(s,uv)} f(s, uv, P) = \rho_{uv} d_{uv} \qquad \text{for all } s \neq 0 \text{ and all } uv \in D_s \;, \tag{7}$$

must be satisfied. The **node-flow constraints**

$$\sum_{P \in \mathcal{P}(0,uv): w \in P} f(0, uv, P) \leq \delta_{uv} d_{uv} \qquad \text{for all } uv \in D \text{ and all } w \in V - \{u, v\} \;, \tag{8}$$

and the **edge-flow constraints**

$$f(0, uv, P) \leq \delta_{uv} d_{uv} \qquad \text{for all } uv \in D \text{ and all } P = \{uv\} \;, \tag{9}$$

are the **diversification constraints**. The summation in the node-flow constraints is over all short paths between nodes u and v that contain node w. These constraints restrict the amount of flow dedicated to a particular demand that goes through a particular node, i.e., they ensure that in the normal operating state, no more than a fraction δ_{uv} of the total demand d_{uv} between nodes u and v flows through a single node w. The node-flow constraints imply that every edge $e \in E$ carries no more than $\delta_{uv} d_{uv}$ of the traffic between u and v, unless $e = uv$. To cover the latter case, the edge-flow constraints are used. These are employed only, of course, if E contains edges between u and v (which are considered as paths $P = \{uv\}$). The constraints (8) and (9) yield that the flow between u and v is diversified, i.e., is routed on at least $\lceil \frac{1}{\delta_{uv}} \rceil$ node-disjoint paths.

The **nonnegativity** constraints

$$f \geq 0 \tag{10}$$

state that all routing variables must have a nonnegative value.

Putting all this together, the mixed-integer linear programming model, denoted by $\mathcal{NDP}$, of our network dimensioning problem is the following:

$$
\begin{array}{lll}
 & \min \sum_{e\in E}\sum_{t=1}^{T_e} k_e^t x_e^t & \\
\text{s.t.} & 1 = x_e^0 \geq x_e^1 \geq \cdots \geq x_e^{T_e} \geq 0 & \text{for all } e \in E \\
 & x_e^t \in \{0,1\} & \text{for all } e \in E \text{ and } t = 1,\ldots,T_e \\
 & y_e = \sum_{t=0}^{T_e} m_e^t x_e^t & \text{for all } e \in E \\
 & y_e \geq \sum_{uv\in D_s} \sum_{P\in\mathcal{P}(s,uv):e\in P} f(s,uv,P) & \text{for all } s \text{ and all } e \in E_s \\
 & \sum_{P\in\mathcal{P}(0,uv)} f(0,uv,P) = d_{uv} & \text{for all } uv \in D \\
 & \sum_{P\in\mathcal{P}(s,uv)} f(s,uv,P) = \rho_{uv} d_{uv} & \text{for all } s \neq 0 \text{ and all } uv \in D_s \\
 & \sum_{P\in\mathcal{P}(0,uv):w\in P} f(0,uv,P) \leq \delta_{uv} d_{uv} & \text{for all } uv \in D \text{ and all } w \in V - \{u,v\} \\
 & f(0,uv,P) \leq \delta_{uv} d_{uv} & \text{for all } uv \in D \text{ and all } P = \{uv\} \\
 & f(s,uv,P) \geq 0 & \text{for all } s, \text{ all } uv \in D_s \text{ and all } P \in \mathcal{P}(s,uv)
\end{array}
$$

A **feasible solution** of our problem is a 0/1-vector x, that satisfies the ordering constraints and yields a **feasible capacity vector** y. The capacity vector y, which is calculated from x using (4), is **feasible** if it permits feasible routings for all operating states, i.e., if the system of linear equations and inequalities (5), ..., (10) has a feasible solution for y.

3 Algorithmic Approach

A close look at the mixed-integer programming formulation $\mathcal{NDP}$ of our network dimensioning problem suggests a decomposition approach for its solution. The problem consists of an "integral part" (deciding the decision variables x) and a "continuous part" (determining the flow variables $f(s,uv,P)$); both parts are linked by the auxiliary variables y. Let us set

$$
Y \quad := \quad \{y \in \mathbb{R}^E \mid \exists f \text{ such that } (y,f) \text{ satisfies } (5),\ldots,(10)\}, \tag{11}
$$

and

$$
X \quad := \quad \operatorname{conv}\{x = (x_e^t)_{e\in E, t=0,\ldots,T_e} \mid \begin{array}{l} x_e^t \in \{0,1\},\ x \text{ satisfies (2), and} \\ y \in Y, \text{where } y_e = \sum_{t=0}^{T_e} m_e^t x_e^t, e \in E\}. \end{array} \tag{12}
$$

We call the integer program

$$(MP) \;\; \min\{k^T x \mid x \in X\},$$

where $k = (k_e^t)_{e\in E, t=0,\ldots,T_e}$, our **master problem**. Since we do not know how to describe the polyhedron X by linear equalities and inequalities we solve, instead of (MP), a relaxation

$$(MP') \ \min\{k^T x \mid Ax \leq b\}$$

where the system $Ax \leq b$ contains the nonnegativity constraints, the ordering constraints (2) and further inequalities valid for X. These further inequalities are chosen from a set of inequalities, to be described later, and are determined algorithmically "on the run" using a cutting plane procedure.

Given a solution x of (MP') we can compute, via (4), a capacity vector y and ask

$$(FP) \text{ Is } y \in Y?$$

Deciding this question is our **feasibility problem**, i.e., solving (FP) means to check whether the (not necessarily integral) capacity vector y, determined through a relaxation (MP') of the master problem (MP) admits feasible routings $f(s, uv, P)$ of the demands in all operating states.

Our algorithmic approach utilizes this way of decomposing the network dimensioning problem as follows. We begin with an initial LP-relaxation $Ax \leq b$ of the master problem and find an optimal solution $\overline{x}$ with the simplex method. We know several classes of inequalities valid for X, see Section 4, and check, using the separation algorithms described in Section 4, whether $\overline{x}$ satisfies these. If not, we add the violated inequalities found to the current LP-relaxation and resolve. If the separation algorithms do not find any violated inequality, we set $\overline{y} := \sum_{t=0}^{T_e} m_e^t \overline{x}_e^t$ and solve the feasibility problem (FP) for $\overline{y}$. This means that we run a sequence of linear programs, one for each operating state.

In case $\overline{y}$ is not feasible for one of the states, then an inequality in the x-variables can be derived that is violated by the current $\overline{x}$. This inequality is added to the current LP-relaxation (MP') and we restart with solving the new relaxation.

If the capacity vector $\overline{y}$ turns out to be feasible for all operating states then there are two possibilities. If $\overline{x}$ is integral we have found an optimal solution of $\mathcal{NDP}$, where the routings are given by the values $f(s, uv, P)$ of the last feasibility test. If $\overline{x}$ is not integral we resort to heuristics to find "reasonable" integral solutions. We could continue with a branch-and-bound procedure, but that turned out to be too time consuming in our practical application.

This process is sketched in the flow chart of Figure 2.

The cutting plane phase provides a lower bound $z_{LP} = \sum_{e\in E} \sum_{t=1}^{T_e} k_e^t \overline{x}_e^t$, where $\overline{x}$ is an optimal solution of the last LP-relaxation (MP'); and the best heuristic solution provides an upper bound z_{IP} to the unknown optimal solution value of (MP). Thus, we get a quality guarantee for the best solution found by the algorithm, i.e., an upper bound on the gap between the values of the best solution found and an optimal solution, given by the quantity $\frac{z_{IP}-z_{LP}}{z_{LP}}100$.

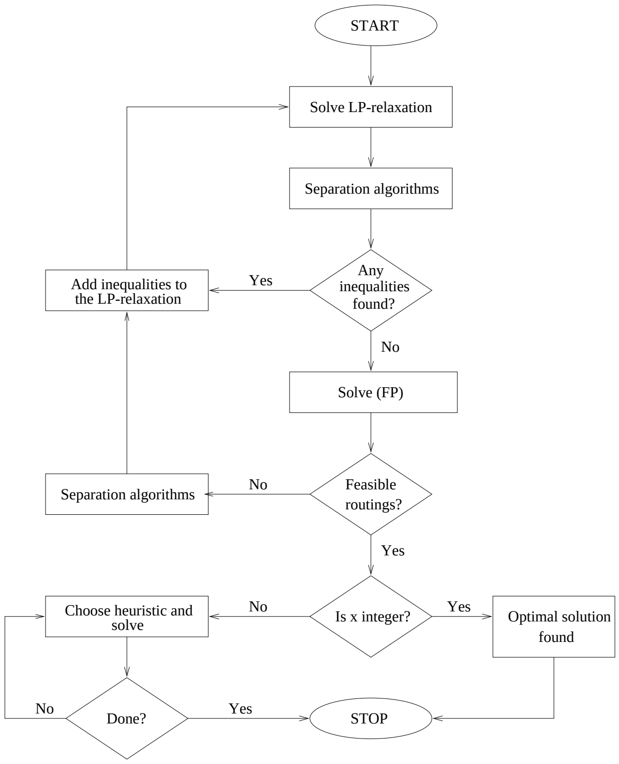

Figure 2: Flow chart of the algorithm.

4 Valid Inequalities

In this section we describe valid inequalities for the polytopes X and Y introduced in Section 3. First we describe necessary and sufficient conditions for a capacity vector y to be in Y, and then we present several classes of inequalities valid for X.

4.1 Valid inequalities for Y

Metric inequalities

Suppose a supply graph $G = (V, E)$ with capacities y_e, for all $e \in E$, and a demand graph $H = (V, D)$ with demands d_{uv}, for all $uv \in D$, are given. The problem of deciding whether, for each $uv \in D$, flow vectors exist such that all demands can be satisfied without violating the capacities is the decision version of a standard multicommodity-flow problem. Iri [Iri71], and Kakusho and Onaga [KO71] have shown that a capacity vector y is feasible for this problem if and only if, for each choice of values $\mu_e \geq 0$ $(e \in E)$, the inequality

$$\sum_{e \in E} \mu_e y_e \geq \sum_{uv \in D} \pi_{uv} d_{uv} \tag{13}$$

is satisfied, where π_{uv} is the shortest $[u, v]$-path value with respect to the weights $\mu_.$. In our case, the multicommodity-flow problems are more complicated in the normal operating state, because of diversification and path-length restrictions. The above result, however, can be modified as follows.

A capacity vector y is feasible for the normal operating state if and only if, for each choice of values $\mu_e \geq 0$ $(e \in E)$, $\gamma^w_{uv} \geq 0$ $(uv \in D, w \in V - \{u,v\})$ and $\gamma^{uv}_{uv} \geq 0$ $(uv \in D, uv \in E)$, the following inequality

$$\sum_{e \in E} \mu_e y_e \geq \sum_{uv \in D} d_{uv}\pi_{uv} - \sum_{uv \in D} (\delta_{uv} d_{uv} \gamma^{uv}_{uv} + \sum_{w \neq u,v} \delta_{uv} d_{uv} \gamma^w_{uv}) \tag{14}$$

is satisfied. Here, π_{uv} is calculated as follows. Given $uv \in D$, we assign to each edge $e \in E - \{uv\}$ the weight μ_e, to edge uv (if it is contained in E) the weight $\mu_e + \gamma^{uv}_{uv}$, and to each node $w \in V - \{u,v\}$ the weight γ^w_{uv}. Then π_{uv} is the value of a shortest among all $[u,v]$-paths with at most ℓ_{uv} edges.

Inequalities (13) and (14) are called **metric inequalities**, see, e.g., [DS94].

Cut inequalities

A special case of a metric inequality is a **cut** inequality. Given $W \subseteq V$, define $\mu_e = 1$ for every $e \in \delta_G(W) = \{e = (w,z) \in E : w \in W,\ z \in V - W\}$, and $\mu_e = 0$ otherwise. Furthermore, set $\gamma^w_{uv} = \gamma^{uv}_{uv} = 0$ for all $uv \in D$, $w \in V - \{u,v\}$. It can be shown that, under appropriate connectivity assumptions, $\pi_{uv} = 1$ for every $uv \in \delta_H(W)$, and $\pi_{uv} = 0$ otherwise, are the shortest $[u,v]$-path values with respect to the edge weights μ. Then inequality (14) reads as follows:

$$\sum_{e \in \delta_G(W)} y_e \geq \sum_{uv \in \delta_H(W)} d_{uv} \ . \tag{15}$$

We call these inequalities **cut inequalities**.

Metric and thus cut inequalities are valid for Y, but, in general they do not define facets of X, when transformed via (4). Applying the same procedure to the induced graphs $G_s = (V_s, E_s)$ and $H_s = (V_s, D_s)$ without the diversification dependent parameters γ^w_{uv} and γ^{uv}_{uv} the inequalities (14) and (15) are easily adapted to failure situations.

4.2 Valid inequalities for X

Based on valid inequalities for Y we now derive two classes of valid inequalities for X. The first class, the *strengthened metric inequalities*, is the result of a divide-and-round procedure. The second class, the *band inequalities*, is similar to minimal cover inequalities for the knapsack problem (see, e.g., Padberg [Pad75]). In our case, Dahl and Stoer [DS92] showed that these can be strengthened, because of the reservation constraints, to the so-called *strengthened band inequalities*. A third class of inequalities for X, that is not based on a valid inequality for Y, is that of *diversification-band inequalities*. If survivability is implemented setting the diversification parameter and not the reservation parameter this third class has proven to be useful in the lower bound calculation.

Strengthened metric inequalities

Let $\sum_{e\in F}\mu_e y_e \geq d$, $F \subseteq E$, be a valid inequality for Y. Using equality (4) we substitute y-variables by x-variables and get the inequality

$$\sum_{e\in F}\mu_e\sum_{t=0}^{T_e} m_e^t x_e^t \geq d\ , \tag{16}$$

which is apparently valid for X. To this inequality we apply a divide-and-round procedure to get a stronger inequality. Let g denote the greatest common divisor of the numbers $\mu_e m_e^t$ for all $e \in F$ and all $t = 0,\dots,T_e$ in inequality (16). Dividing the coefficients of (16) by g we get, due to the integrality of every feasible solution, the **strengthened metric inequality**

$$\sum_{e\in F}\mu_e\sum_{t=0}^{T_e}\frac{m_e^t}{g}x_e^t \geq \left\lceil\frac{d}{g}\right\rceil\ . \tag{17}$$

Regarding the LP's to solve we try to avoid the strengthening of metric inequalities. These inequalities are very dense (every breakpoint of every supply edge in the set F appears in the strengthened metric inequality) and have "wild" coefficients, and thus, they may cause numerical instabilities in the course of solving the LP.

Band inequalities

Let $\sum_{e\in F}\mu_e y_e \geq d$, $F \subseteq E$, be a valid inequality for Y. Assign to every supply edge $e \in F$ a breakpoint t_e $(0 \leq t_e < T_e)$ such that $\sum_{e\in F}\mu_e M_e^{t_e} < d$. Then the **band inequality**

$$\sum_{e\in F} x_e^{t_e+1} \geq 1 \tag{18}$$

is valid for X. Dahl and Stoer [DS94] have shown that maximal band inequalities are, under rather weak conditions, facet defining for the polytope

$$\mathrm{conv}\{x\in\{0,1\}^{|T(F)|} \mid \sum_{e\in F}\mu_e\sum_{t=0}^{T_e} m_e^t x_e^t \geq d,\ 1 = x_e^0 \geq x_e^1 \geq \cdots \geq x_e^{T_e} \geq 0, e\in F\}\ , \tag{19}$$

where $|T(F)| = \sum_{e\in F} T_e$; see also Wolsey [Wol90]. Band inequalities can be strengthened in the case of edge failures. Let $d = \sum_{uv\in D}\pi_{uv}d_{uv}$ in inequality (16) and, as before, assign to every supply edge $e \in F$ a breakpoint t_e $(0 \leq t_e < T_e)$. If $\sum_{e\in F-\{\overline{e}\}}\mu_e M_e^{t_e} < \sum_{uv\in D_{\overline{e}}}\pi_{uv}\rho_{uv}d_{uv}$ for every $\overline{e} \in F$, then we derive the inequality

$$\sum_{e\in F} x_e^{t_e+1} \geq 2\ , \tag{20}$$

which is valid for X. We refer to (20) as a **strengthened band inequality**.

Diversification-band inequalities

The third class of valid inequalities for the polytope X is based on diversification. The diversification parameter bounds, for every demand edge, the corresponding flow through any component. Applying this to the edges of a cut yields diversification-band inequalities.

Let $W \subseteq V$ be a subset of the nodes and let us define the following quantities:

$d = \sum_{uv \in \delta_H(W)} d_{uv}$,	the demand that has to use the supply edges $\delta_G(W)$,
$\alpha = \sum_{uv \in \delta_H(W)} \delta_{uv} d_{uv}$,	the maximum fraction of d that any of the supply edges in $\delta_G(W)$ can serve.

Now, we assign to every supply edge $e \in \delta_G(W)$ some breakpoint t_e, $0 \leq t_e < T_e$, together with its capacity $M_e^{t_e}$. Given these chosen capacities, for every feasible flow in the supply graph G, the flow through a supply edge $e \in \delta_G(W)$ supplying part of the demand d is at most $\min\{M_e^{t_e}, \alpha\}$. In case the "remaining demand"

$$r := d - \sum_{e \in \delta_G(W)} \min\{M_e^{t_e}, \alpha\}$$

is positive we have to increase the chosen capacities to satisfy d. Since every edge can carry at most a flow of value α, at least $\lceil \frac{r}{\alpha} \rceil$ of the incremental capacities $m_e^{t_e+1}, e \in \delta_G(W)$, have to be chosen in addition. Hence, for each choice of breakpoints t_e, $0 \leq t_e < T_e$, $(e \in \delta_G(W)$, the inequality

$$\sum_{e \in \delta_G(W)} x_e^{t_e+1} \geq \left\lceil \frac{r}{\alpha} \right\rceil$$

is valid for X.

4.3 Separation of inequalities

The separation problem for the metric inequalities can be solved in polynomial time using linear programming. Whenever we test the feasibility of a capacity vector we find a violated metric inequality if the provided capacities are not feasible. Details follow in Section 5.

However, the separation problem for band and strengthened band inequalities is equivalent to the $\mathcal{NP}-hard$ multiple-choice knapsack problem. To solve this separation problem we use a heuristic algorithm proposed by Dahl and Stoer [DS92], and an exact algorithm based on a dynamic programming algorithm for the multiple-choice knapsack problem (see Martello and Toth [MT90]).

For the separation of diversification-band inequalities, we have implemented several heuristics whose tedious details we do not want to describe here. Note that a diversification-band inequality is determined by a cut $\delta_G(W)$ and a choice of breakpoints $t_e, e \in \delta_G(W)$. In each of our heuristics we concentrate on a limited set of cuts (e.g., cuts found in previous runs of the cut inequality separation procedure, or cuts with small shores) and choose breakpoints by restricted enumeration of "promising" breakpoint combinations.

5 The multicommodity feasibility problem

Having computed a vector x and the corresponding capacity vector y with components $y_e = \sum_{t=0}^{T_e} m_e^t x_e^t$, one has to test whether the capacities satisfy the various requirements of the network in all operating states, i.e., one has to solve the feasibility problem (FP). This amounts to solving as many multicommodity-flow problems as there are operating states. Although these tests can be done using fast LP-solvers, the overall time to check feasibility is high. Therefore, whenever it is possible to infer the feasibility or infeasibility of the given capacity vector by faster means this should be done.

5.1 Alternative ways to decide feasibility of the capacity vector

Given a capacity vector y we apply various simple tests to determine whether y provides feasible routings for some operating states. Given feasible routings in the normal operating state, we denote by $flow(e)$ the flow through e for every supply edge $e \in E$, and by $flow_{uv}(w)$ the part of the demand d_{uv} that is routed through w for every demand edge uv and every node $w \in V - \{u, v\}$. Both values, $flow(e)$ and $flow_{uv}(w)$ are easily calculated using the inequality 8 and 9. respectively.

If one of the following criteria is satisfied we can skip solving the multicommodity-flow problem for the respective operating state, since the routings provided in the normal operating state remain feasible in this state.

Criterion 1. Assume that the capacities in the normal operating state are feasible and consider a supply edge $e \in E$. If for all demands $uv \in D$ the inequality $d_{uv} - flow(e) \geq \rho_{uv} d_{uv}$ holds, then there are feasible routings if edge e fails.

Criterion 2. Assume that the capacities in the normal operating state are feasible and consider a node $w \in V$. If for all demands $uv \in D_w$ the inequality $flow_{uv}(w) \leq (1 - \rho_{uv}) d_{uv}$ holds, then there are feasible routings if node w fails.

Even though very simple, these criteria apply quite often, particularly in the decrease heuristics.

5.2 Formulation of the feasibility problems

As we mentioned above, if there are no other means to determine whether a given capacity vector y is feasible or not one has to solve the multicommodity-flow problems. There is one more reason one would be willing to do so. What is actually needed is not only an answer whether the capacity vector is feasible or not. If it is not feasible, some inequality valid for the polytope X needs to be generated such that, when added to the LP-relaxation (MP'), it will cut off the current nonfeasible solution point x. It turns out that one can formulate the feasibility problems in such a way that such an inequality can be derived whenever the problem is not feasible; see Minoux [Min81].

We introduce a new variable α that has the following meaning. If an additional capacity α is added to each capacity y_e then feasible routings exist. Our goal is to make α as small as possible. If the minimum value is positive the capacity vector is not feasible, otherwise it is. The multicommodity-flow problem for a particular failure state s, can, thus, be viewed as follows:

$$
\begin{aligned}
\min \quad & \alpha && && (21)\\
\text{s.t.} \quad & \sum_{uv \in D_s} \sum_{P \in \mathcal{P}(s,uv): e \in P} f(s,uv,P) - \alpha \le y_e && \text{for all } e \in E_s && (22)\\
& \sum_{P \in \mathcal{P}(s,uv)} f(s,uv,P) = \rho_{uv} d_{uv} && \text{for all } uv \in D_s && (23)\\
& f(s,uv,P) \ge 0 && \text{for all } uv \in D_s,\ P \in \mathcal{P}(s,uv) && (24)
\end{aligned}
$$

In the case of diversification, one has to consider in the normal operating state the diversification constraints8, 9 in addition to the ones above.

The LP-dual of the above multicommodity-flow problem is the following:

$$
\begin{aligned}
\max \quad & \sum_{uv \in D_s} \rho_{uv} d_{uv} \pi_{uv} - \sum_{e \in E_s} y_e \mu_e && && (25)\\
\text{s.t.} \quad & -\sum_{e \in P} \mu_e + \pi_{uv} \le 0 && \text{for all } uv \in D_s, P \in \mathcal{P}(s,uv) && (26)\\
& \sum_{e \in E_s} \mu_e = 1 && && (27)\\
& \mu_e \ge 0 && \text{for all } e \in E_s && (28)
\end{aligned}
$$

Two observations are in order. First, from linear programming duality we get that if the capacity vector y is infeasible, in which case the optimal value α^* of the primal feasibility problem is positive, then y satisfies

$$
\sum_{uv \in D_s} \rho_{uv} d_{uv} \pi_{uv} > \sum_{e \in E_s} y_e \mu_e \ ,
$$

i.e., y violates a metric inequality, see Section 4. Therefore solving the feasibility problem we automatically generate a violated metric inequality whenever the problem is not feasible. The second observation is that the feasibility problem can be solved using a column generation approach in which the auxiliary problem is a restricted shortest path problem, see Section 5.4.

5.3 Solving the feasibility problems

The variables of the feasibility problem correspond to the valid paths for each demand. This number is in general exponential and thus we follow a column generation approach to solve the linear programs corresponding to the feasibility problems.

We solve the primal feasibility problem for a "well-chosen" subset of the variables (valid paths) and get the optimal value α^* and the primal and dual variables. Clearly, if $\alpha^* \leq 0$, we can stop since we know that the true optimal value involving all variables is at most as large as the one obtained with the restricted number of variables. If $\alpha^* > 0$ then we need to continue in order to find the optimal value of α or decide that the current one is optimal. To this end, we solve, for each demand, the restricted shortest path problem, see Section 5.4, with the dual variables μ_e as weights on the edges. If the restricted shortest path length is smaller than the value of the dual variable π_{uv} of the particular demand then this path violates one of the constraints (26) and thus the corresponding variable is added to the set of primal variables. If the length of the restricted shortest path is larger than the value π_{uv}, for all demands $uv \in D_s$, then the current solution α^* is optimal.

5.4 Solving the restricted length shortest path problem

The auxiliary problem of the column generation procedure described above is a restricted shortest path problem. In general the problem is defined as follows. Given a simple graph $G = (V, E)$, a node $u \in V$, weights $\mu_e \geq 0$ and lengths $\lambda_e \geq 0$ for each edge $e \in E$, we want to find the minimum-weight path of length at most ℓ_{uv} from u to every other node v. The problem of deciding whether a path of weight at most M and length at most L exists between two specified nodes is in general $\mathcal{NP}-complete$; however, it is polynomially solvable if all the weights or all lengths are equal; see Garey and Johnson [GJ79]. In our case, $\lambda_e = 1$ for all $e \in E$ and thus the problem is polynomially solvable.

The algorithm to solve the restricted shortest path problem is a modification of the Dijkstra shortest path algorithm, [Dij59]. Let ℓ denote the value of the path length restriction and suppose that we want to calculate the length restricted shortest path tree for a node $u \in V$. In every iteration of the algorithm we determine the length restricted shortest path for exactly one node. This node will be called *labeled*. The algorithm terminates if all nodes are labeled that can be reached from u on a path with at most ℓ edges.

In more detail, for every $v \in V$, let $d_i(v)$ be the shortest distance from u to v using at most i edges in the current iteration of the algorithm. Denote by U be the set of all unlabeled nodes, initially set to $V - \{u\}$, and define $R := \{v \in U : \exists k \leq \ell \text{ with } d_k(v) < \infty\} \subseteq U$; i.e., R is the set of unlabeled nodes that can be reached from u using only labeled intermediate nodes. At each iteration of the algorithm, we label the node with the smallest distance from u, breaking ties by selecting the one corresponding to the path with the fewest edges, and then we update the distance labels $d_i(v)$ of its unlabeled neighbors.

```
algorithm RESTRICTED SHORTEST PATH
begin
   U := V − {u}, R := {v ∈ V | (u, v) ∈ E}
   For each v ∈ V set
              ⎧ 0    if v = u
      d_i(v) := ⎨ μ_e  if e = (u, v) ∈ E
              ⎩ ∞    otherwise
   while R ≠ ∅
      find v ∈ R and 1 ≤ i ≤ ℓ such that ∀w ∈ R, 1 ≤ j ≤ ℓ
         (i) d_i(v) ≤ d_j(w) and (ii) d_i(v) = d_j(w) ⇒ j ≥ i
      U := U − {v}, R := R − {v}
      for all neighbors w ∈ U
         for all j with i + 1 ≤ j ≤ ℓ
            d_j(w) = min{d_j(w), d_{j−1}(v) + μ_(v,w)}
         end
         if d_ℓ(w) < ∞
            R := R ∪ {w}
         end
      end
   end
end
```

The restricted length shortest paths from u to $v \in V - \{u\}$ can be easily determined by keeping track of the predecessors for each $v \in V$ and each $1 \leq i \leq \ell$.

6 Heuristic algorithms

In this section we present two classes of heuristics we use to get integer feasible solutions: the decrease and the increase heuristics. Since none of the heuristics outperforms consistently the others, in a typical run of DISCNET we run all of them.

6.1 Decrease Heuristics

In the decrease heuristics we start with a feasible capacity vector and try to reduce the capacity of its components (supply edges) keeping it feasible. A feasible capacity vector is obtained by rounding up the capacity of each edge – as calculated from the solution of the LP-relaxation

(MP') via (4) – to the next breakpoint capacity. One can select the particular edge for which the capacity reduction will be tried, and the number of reductions to be tried on this particular edge. By reduction we mean reducing the capacity of an edge from the current breakpoint capacity to that of a smaller breakpoint capacity.

We consider the following five criteria for the selection of the edge and the capacity reduction strategy, that give rise to five decrease heuristics.

Criterion 1. Select the edge with the smallest fractional x_e^t, and reduce its capacity as much as possible.

Criterion 2. Select the edge whose capacity reduction will incur the biggest cost reduction, and reduce its capacity to the previous breakpoint capacity.

Criterion 3. Select the edge whose capacity reduction will incur the biggest cost reduction, and reduce its capacity as much as possible.

Criterion 4. Select the edge whose reduction will incur the biggest relative cost reduction (cost reduction per unit of capacity reduction), and reduce its capacity as much as possible.

Criterion 5. Select the edge whose reduction will incur the biggest relative cost reduction (cost reduction per unit of capacity reduction), and reduce its capacity to the previous breakpoint capacity.

6.2 Increase Heuristics

The increase heuristics are partial branch-and-cut heuristics. In particular, if the current x-vector has fractional components, we select one of them to be fixed at the value of 1 and then we proceed with the cutting plane algorithm. It should be noted that although we fix only one breakpoint at a time, probably more are fixed implicitly due to the ordering constraints (2). The outcome of an increase heuristic can be either a feasible integer solution or an infeasible integer solution. In the second case, we try to restore feasibility using the feasibility tests (which, of course, will return some violated metric inequality since the current x-vector is not feasible). In both cases, we apply the decrease heuristics in an effort to reduce the obtained solution cost.

In the increase heuristics one has to choose a decision variable x_e^t, i.e., a particular edge and a particular breakpoint of that edge, to fix. The candidates are those variables that satisfy $0 < x_e^t < 1$. We have developed the following four criteria for such a selection which give rise to four increase heuristics.

Criterion 1. Choose the fractional variable x_e^t closest to 1.

Criterion 2. Choose the fractional variable x_e^t that will incur the minimum additional cost $(1 - x_e^t)k_e^t$.

Criterion 3. Choose the fractional variable x_e^t such that the $M_e^t - y_e$, i.e., the additional capacity needed to make y_e a breakpoint capacity, is minimal.

Criterion 4. Choose the fractional variable x_e^τ such that $K_e^\tau - \sum_{t=1}^{T_e} k_e^t x_e^t$, i.e., the additional cost incurred from increasing y_e to the breakpoint capacity M_e^τ, is minimal.

7 Computational Results

In this section we present computational results for different problem instances supplied by `e-plus`, with various settings for the diversification, reservation and path-length parameters.

The characteristics of the three networks we use are given in Table 1. The unit of the demand value is a channel (64 kbit/s). The available capacities for each supply edge are multiples of 30 channels (2 Mbit/s), multiples of 480 channels (34 Mbit/s) and multiples of 1920 channels (140 Mbit/s), and any combination of these three capacities. Clearly, one has to choose a "reasonable" number of capacities (breakpoints) for each supply edge, since the problem size gets too big if all possible capacity combinations for each supply edge are considered. We consider 3, 5, and 7 breakpoints with the capacities shown in Table 2.

Name	$\|V\|$	$\|E\|$	$\|D\|$	range of demands
Network1	11	34	24	95 – 384
Network2	12	53	28	34 – 480
Network3	14	39	82	30 – 360

Table 1: Characteristics of the test problems.

No. of breakpoints	Capacities (in channels)						
3	30	480	960				
5	30	60	90	480	960		
7	30	60	90	120	150	480	960

Table 2: Breakpoint capacities.

The parameters are set as follows. The path-length parameter takes the values 3, 5, and ∞ (no length restriction). For each of these values, four different diversification/reservation settings are tested. Table 3 shows these parameter settings together with the names of the respective problems.

Name	d1r0	d1r50	d1r100	d50r0
Diversification	1.0	1.0	1.0	0.5
Reservation	0.0	0.5	1.0	0.0

Table 3: Diversification and reservation parameter settings.

The total time reported in Table 4 corresponds to a *complete run* of the program that consists of the calculation of the lower bound (cutting-plane part), the execution of five decrease heuristics, and the execution of four increase heuristics. All runs were done on a `SPARCSTATION 20` with a `71 SuperSPARC-II` processor and 192MB RAM. Each of the increase heuristics is followed by a run of all decrease heuristics. Keeping this in mind, the times shown in Table 4 are reasonable, considering also the fact that DISCNET is a tool to be used during the planning process, which typically starts long time before the realization of the network.

Name	ℓ	d1r0	d1r50	d1r100	d50r0
Network1	3	0:21 - 4:27	4:06 - 12:08	5:29 - 12:15	2:39 - 8:40
	5	0:21 - 4:50	5:11 - 13:57	5:59 - 14:04	3:17 - 8:36
	∞	0:21 - 4:18	4:41 - 14:26	5:36 - 13:59	3:55 - 8:56
Network2	3	2:10 - 16:15	23:26 - 84:47	26:00 - 122:01	11:53 - 37:18
	5	1:47 - 16:47	35:43 - 89:34	26:44 - 127:57	12:27 - 39:00
	∞	1:45 - 16:37	36:30 - 93:55	29:39 - 130:57	13:14 - 42:45
Network3	3	2:31 - 25:06	28:19 - 81:03	32:30 - 72:16	29:40 - 75:35
	5	3:05 - 31:37	42:18 - 93:17	46:00 - 82:14	54:11 - 95:44
	∞	2:38 - 29:06	42:50 - 110:38	53:30 - 98:39	53:52 - 94:27

Table 4: Computation time ranges (in min:secs).

Name	ℓ	d1r0	d1r50	d1r100	d50r0
Network1	3	38 - 45	36 - 41	24 - 26	35 - 41
	5	39 - 41	30 - 31	19 - 22	31 - 32
	∞	39 - 41	26 - 29	18 - 22	30 - 31
Network2	3	34 - 43	32 - 33	21 - 22	23 - 31
	5	37 - 41	29 - 31	19 - 20	23 - 24
	∞	37 - 41	27 - 30	19 - 20	21 - 26
Network3	3	46 - 49	29 - 33	26 - 27	40 - 43
	5	43 - 46	31 - 33	24 - 25	41 - 46
	∞	44 - 48	30 - 32	23 - 24	38 - 41

Table 5: Gap ranges(%).

From Table 5 we see that the gaps are quite large. We believe that the reason for this is the weak lower bound, which in turn is a result of the restricted knowledge of the facial structure of the polytope X. In particular, the smaller the reservation parameter, the large the gap.

8 Implementing Survivability

In this section we compare and discuss three ways to introduce survivability at the transport layer by appropriately setting parameters of the presented model.

The physical network is said to have survivability of $\alpha\%$, if at least $\alpha\%$ of each demand can be satisfied in case of a single node or single edge failure. In our model we have two input parameters which are used to introduce survivability; the diversification parameter and the reservation parameter. These two parameters can be set one at a time or in any combination.

Setting the diversification parameter δ_{uv} for the demand d_{uv} of the logical switching network, we require that at most $100\delta_{uv}\%$ of d_{uv} is routed through any node (other than u and v) or any link of the physical network. This implies that we get routings which provide node disjoint paths, each of them carrying at most $\delta_{uv}d_{uv}$ channels, and therefore, only that many channels of the demand can be lost in a single node or single link failure. That is, $(1-\delta_{uv})d_{uv}$ channels "survive" without any rerouting effort. There are two drawbacks, however. First, setting the diversification parameter to δ_{uv} implies that the demand will be routed through at least $\lceil \frac{1}{\delta_{uv}} \rceil$ node disjoint paths. For example, setting δ_{uv} to 0.49, we require at least 3 node disjoint paths on which we route d_{uv}. Second, we cannot achieve 100% survivability with this parameter, and diversification values below $\frac{1}{3}$ are undesirable by the network operator, because this would force at least 4 paths each of them carrying only a small fraction of the demand. Another drawback is the high cost of the resulting network; see Figure 3.

Using the reservation parameter to introduce survivability we take advantage of possible redundancy in the network by allowing rerouting in failure situations. Depending on the particular failure all demands might be rerouted. For a specific demand of d_{uv} channels the reservation parameter ρ_{uv} guarantees that at least $\rho_{uv}d_{uv}$ channels will be still satisfied in a failure state. In our tests we have observed that much more can actually be satisfied. For instance, by maximizing – in a post processing step – the total satisfied demand, we found that all but few demands are indeed fully satisfied. However, it should be noted that this is an empirical observation, and in theory one can guarantee only that $\rho_{uv}d_{uv}$ channels will "survive" a failure. The advantage of this method is the design of low cost networks; see Figure 3, also for a comparison with the costs of the previous method. The obvious disadvantage of this method is the need for rerouting in case of a failure. Indeed, as we have observed in practice, this rerouting may be extensive which makes the management of the network rather difficult.

To compare the costs of the two methods, we made several test runs for *Network1*; see Section 7. We choose as survivability values 0%, 25%, 50%, 66%, 75% and 100%, where the last value cannot be achieved using the diversification parameter only. Although 75% survivability can be achieved by setting the diversification parameter equal to 0.25, we do not consider this option because, as we mentioned above, this forces too many paths for each demand. For the other values of survivability, the corresponding diversification/reservation values are 1.0/0.0, 0.75/0.25, 0.5/0.5 and 0.34/0.66. The best solution values we get with our network dimensioning tool DISCNET are shown in Figure 3.

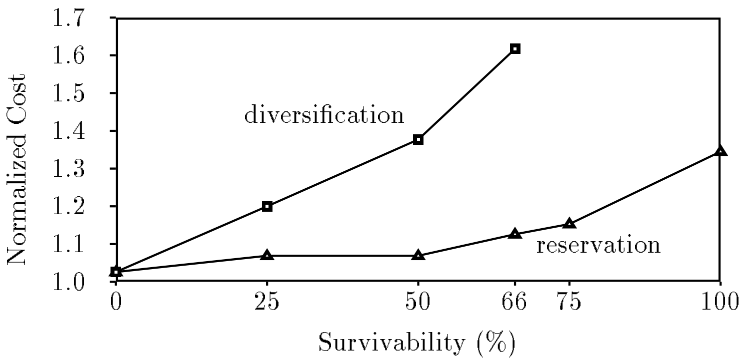

Figure 3: Comparison of costs of introducing survivability by setting the diversification or the reservation parameters

In general, there is a trade-off between the easiness of the network management provided by the first method and the total installation cost provided by the second. Since the network management costs are not included in the installation costs, it is up to the network operator to decide whether these costs counterbalance the difference in the installation costs.

A third way we consider to introduce survivability in the network, is a combination of the two methods. A minimum survivability is achieved by the diversification parameter setting, with the advantage of easy network management. Additional survivability is introduced by the reservation parameter setting. In case of a failure situation the operator has to decide whether to reconfigure the network, or not. This decision depends on various aspects, e.g., on the affected traffic, the expected recovery time, and the required effort to reconfigure the network.

To compare the cost of implementing the third method, to those of the previous ones, we run two additional series of tests, combining diversification and reservation parameters. In the first series we keep a minimum survivability of 25% (achieved by setting the diversification parameter to 0.75) and we increase survivability by setting the reservation parameter to 0.50, 0.66, 0.75 and 1.0. In the second series we change the minimum survivability value to 50% and increase survivability by setting the reservation parameter to 0.66, 0.75 and 1.0. We only consider reservation parameter settings bigger than the minimum survivability, since the diversification parameter setting dominates the other cases.

The best solution values we get with DISCNET are shown in Figure 4. The lowest curve in Figure 4 is the reservation curve of Figure 3 (minimum survivability of 0%).

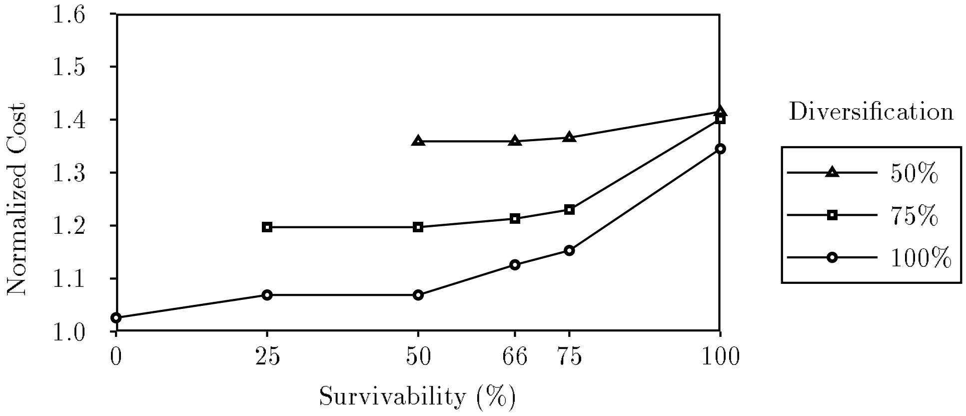

Figure 4: Comparison of costs for the different methods of introducing survivability

9 Conclusions

In this paper we modeled the problem of designing survivable telecommunications networks, and presented an algorithm to solve it. The algorithm is in the core of a network designing tool we developed for `e-plus` Mobilfunk GmbH, called DISCNET. The tool has been implemented in C++ and it is already being used at `e-plus`.

DISCNET solves problem instances of practical interest in reasonable times. The solutions produced for problems with "real" data were 15-20% better than the ones produced by the network designers. It is clear, that designing the network by hand, the network designer cannot incorporate all the restrictions included in our model and satisfied by the solutions produced by DISCNET. Moreover, the sizes of today's networks have grown beyond the size that is manageable by the network designers, making the use of a tool like DISCNET necessary. The output of a typical run, as reported in Section 7, suggests different topologies, including the routings for all operating states. The network designer chooses a topology suitable for his needs, after considering the trade off between high installation cost and maintenance difficulty. As we mentioned in Section 7 the gap is still large, as is often the case in capacitated fixed-charge network design problems; see e.g. Magnanti, Mirchandani and Vachani [MMV95]. We believe that the main reason for this is the poor quality of the calculated lower bound, which in turn is a result of rather weak inequalities used to strengthen the formulation. Therefore, further investigation of the polytope X is necessary.

We conclude with some open problems that are interesting from both a practical and a theoretical point of view.

First, is the problem of finding alternative ways to deal with survivability. The methods discussed in Section 8, impose rather strong restrictions. For example alternative ways could be

- set the percentage of the flow on a particular edge that should be rerouted when the edge or one of its end-nodes fails, or,
- set the percentage of the demand that should be rerouted if one of the edges or nodes that are used to route the demand fails.

Second, is the problem of providing routings that make the network management easier. In our model, besides the diversification constraints, we do not impose any restriction on the routings. We merely guarantee that with the chosen capacities there *exist* feasible routings for every operating state. The network operator, however, might have some preferences that make the network management easier. Such preferences are for instance

- minimal rerouting in the case of failures,
- a maximum number of paths used to route a particular demand (recall that the diversification parameter imposes a minimum number of paths),
- a minimum value for the positive fraction of a demand routed on each path, and
- a maximum number of paths that use a particular link.

None of these requirements is considered in our model. In fact, we believe that adding all these restrictions to the feasibility problems, will make the whole problem intractable. What would be of interest, is to try to satisfy these requirements as much as possible in a post-processing step.

References

[BG95] D. Bienstock and O. Günlük. Computational experience with a difficult mixed-integer multicommodity flow problem. *Mathematical Programming*, 68:213–237, 1995.

[Dij59] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[DS92] G. Dahl and M. Stoer. MULTISUN – Mathematical model and algorithms. Technical Report TF R 46/92, Televerkets Forskningsinstitutt, 1992.

[DS94] G. Dahl and M. Stoer. A polyhedral approach to multicommodity survivable network design. *Numerische Mathematik*, 68:149–167, 1994.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to Theory of NP-completeness.* Freeman, San Francisco, 1979.

[GMS92a] M. Grötschel, C.L. Monma, and M. Stoer. Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research*, 40(2):309–330, 1992.

[GMS92b] M. Grötschel, C.L. Monma, and M. Stoer. Facets for polyhedra arising in the design of communication networks with low-connectivity constraints. *SIAM Journal on Optimization*, 2(3):474–504, 1992.

[Iri71] M. Iri. On an extension of the maximum-flow minimum-cut theorem to multicommodity flows. *Journal of the Operations Research Society of Japan*, 13:129–135, 1971.

[KO71] O. Kakusho and K. Onaga. On feasibility conditions of multicommodity flows in networks. *Transactions on Circuit Theory*, 18:425–429, 1971.

[LSV95] A. Lisser, R. Sarkissian, and J.P. Vial. Optimal joint syntheses of base and spare telecommunication networks. Technical report, University of Genève, November 1995.

[Min81] M. Minoux. Optimum synthesis of a network with non-simultaneous multicommodity flow requirements. In P. Hansen, editor, *Studies on Graphs and Discrete Programming*, pages 269–277. North–Holland Publishing Company, 1981.

[MMP90] C.L. Monma, B.S. Munson, and W.R. Pulleyblank. Minimum-weight two-connected spanning networks. *Mathematical Programming*, 46:153–171, 1990.

[MMV93] T.L. Magnanti, P. Mirchandani, and R. Vachani. The convex hull of two core capacitated network design problems. *Operations Research*, 60(2):233–250, 1993.

[MMV95] T.L. Magnanti, P. Mirchandani, and R. Vachani. Modeling and solving the two–facility capacitated network loading problem. *Operations Research*, 43(1):142–156, 1995.

[MS89] C.L. Monma and D.F. Shallcross. Methods for designing communications networks with certain two-connected survivability constraints. *Operations Research*, 37(4):531–541, 1989.

[MT90] S. Martello and P. Toth. *Knapsack Problems – Algorithms and Computer Implementations*. Discrete Mathematics and Optimization. Wiley, 1990.

[Pad75] M.W. Padberg. A note on zero-one programming. *Operations Research*, 23(4):833–837, 1975.

[Sto92] M. Stoer. *Design of Survivable Networks*. Lecture Notes in Mathematics. Springer, 1992.

[Wol90] L.A. Wolsey. Valid inequalities for 0–1 knapsack and mips with generalized upper bound constraints. *Discrete Applied Mathematics*, 29:251–261, 1990.