UNIVERSITY OF OSLO
Department of informatics

A Cutting Plane Algorithm for Multicommodity Survivable Network Design Problems

Geir Dahl and
Mechthild Stoer

Preprint 3

May 9, 1995

A Cutting Plane Algorithm for Multicommodity Survivable Network Design Problems

Geir Dahl * and Mechthild Stoer †

May 9, 1995

Abstract

We present a cutting plane algorithm for solving the following network design problem in telecommunications: *given* point-to-point traffic demands in a network, specified survivability requirements and a discrete cost/capacity function for each link, *find* minimum cost capacity expansions satisfying the given demands. The algorithm is based on the polyhedral study in the accompanying paper [16]. We describe the underlying problem, the model and the main ingredients in our algorithm: initial formulation, feasibility test, separation for strong cutting planes and primal heuristics. Computational results for a set of real-world problems are reported.

1 Introduction

The design of cost-efficient telecommunications networks meeting requirements concerning traffic, flexibility, survivability etc. is a major challenge with great economic impact. In particular, it is important to establish networks that are robust with respect to accidents like cable cuts, electronic failures or power supply shut-down. Often, the

*Institute of Informatics, University of Oslo, P.O.Box 1080, Blindern, 0316 Oslo, Norway (Email:geird@ifi.uio.no)

†Telenor Research, P.O.Box 83, N-2007 Kjeller, Norway. Email: stoer@nta.no

capacity limitations play a crucial role in these design problems (e.g., capacities of terminal equipment installed at both end nodes of each transmission links). This calls for models and methods for designing low-cost capacitated networks that allow routing of traffic in both normal and specified failure situations. The model we discuss in this paper falls into this framework.

In MULTISUN (MULTIcommodity SUrvivable Network design) we integrate the problems of topological design, capacity assigment and routing. Due to its generality this is a very difficult problem with NP-hard problems as special cases. The main purpose of this paper is to describe a cutting plane algorithm for solving MULTISUN problems and report computational results for some real-world problems of interest.

In the accompanying paper [16] we presented a theoretical study of the MULTISUN problem and identified several classes of facet defining inequalities for certain associated integral polyhedra. We therefore refer to [16] for validity and facet proofs for the inequalities discussed later.

The MULTISUN problem can be described more precisely as follows. Let V be a given set of nodes with traffic demands between certain pairs of these nodes. Each demand represents a certain amount of point-to-point traffic to be routed in the network between the origin and destination nodes. Traffic may be split on several paths, so it may be viewed as a continuous network flow. In addition, we have given "supply" edges joining pairs of nodes in V; these represent existing or potential direct physical links (e.g., a fiber cable or a radio relay system). For each supply edge one wants to decide which capacity to install, selected from a discrete set of alternatives, each with an associated building cost. We are interested in cost-optimal capacity extensions that satisfy the following conditions:

(i) in case of a node or edge failure all demands can be routed simultaneously,

(ii) when all nodes and edges are operating, then all demands can be routed simultaneously such that no more than a given fraction of the given demand is routed through any intermediate node.

A large amount of work has been done by Minoux and others on the related model with a **continuous** cost function, see [12] (and its references), [7], [1]. A recent related model is studied in [3].

This paper is organized as follows. In Section 1 the integer lin-

ear programming model for the MULTISUN problem is presented. In addition it is explained how one obtains stronger LP relaxations by adding certain classes of valid inequalities originating from knapsack-like substructure of the original model. Next, in Section 2, we explain the main components of our cutting plane algorithm for the MULTISUN problem, including separation algorithms and primal heuristics. This algorithm is being used for network planning in Telenor Research. Results for some real-world problems are reported and discussed in Section 3. Finally, in the concluding section, some directions for further work is pointed out.

We use fairly standard notation from graph theory and polyhedral theory, see [2, 14], but a few notions need to be explained. $\mathbf{R}^M$ denotes the space of real vectors indexed by M (where M is some finite set), and for $x \in \mathbf{R}^M$ and $S \subseteq M$ we let $x(S)$ denote $\sum_{i \in S} x_i$. By $\chi^S \in \mathbf{R}^M$ we denote the incidence vector of S, and $\mathbf{1}$ is a suitable dimensioned vector with 1's. Let $G = (V, E)$ be an undirected graph without loops and multiple edges. If $W \subseteq V \cup E$, we let $G - W$ denote the graph obtained from G by removing from G each node in W with their incident edges. The **cut** $\delta_G(W)$ induced by a subset W of V is the set of edges with one end node in W and the other outside W. By $G[W] = (W, E(W))$ we denote the graph induced by node set W. For two nodes u and v, a $[u, v]$-path P is a sequence of consecutive nodes and edges connecting u and v without repeating any nodes. A graph G is said to be **2-edge** (or **2-node**) **connected** with respect to some given node set R, if between any two nodes $u, v \in R$ there exist at least two edge- (or node-) disjoint $[u, v]$-paths. We do not allow G to have parallel edges. A **network** $\mathcal{N} = (G, c)$ is a graph G with weights (e.g., capacities or demands) $c_e \geq 0$ associated with each edge e. Given a **supply** network (G, c) and a **demand** network (H, d), where $G = (V, E)$ and $H = (V, F)$ have the same node set, a **multicommodity flow** (w.r.t. (H, d)) is defined as a collection of $[u, v]$-paths P^i_{uv} in G together with numbers $\lambda^i_{uv} \geq 0$ for each $uv \in F$ and i such that $\sum_i \lambda^i_{uv} = d_{uv}$, for each $uv \in E(H)$. The associated **uv-flow** is the vector $z^{uv} \in \mathbf{R}^E$ with e'th component given by $z^{uv}_e = \sum_{i: e \in P^i_{uv}} \lambda^i_{uv}$ (called the uv-flow in edge e). The network (G, c), or the capacity vector c, is said to **allow** a multicommodity flow w.r.t. (H, d), if $\sum_{uv \in D} z^{uv}_e \leq c_e$ for each $e \in E$, i.e., the total flow in each edge does not exceed its capacity.

2 Mathematical model and improved formulations

In this section we first present the mathematical model for the MULTISUN problem and discuss multicommodity flow requirements in some detail. Next we briefly describe the polyhedral approach to this problem and how it leads to some stronger LP formulations of the problem.

2.1 The MULTISUN model

Each edge of the **supply** graph $G = (V, E)$ corresponds to a physical (transmission) link that has been or can be established. The nodes correspond to switching points. We assume that G is connected (otherwise the problem would decompose). In the **demand** graph $H = (V, D)$ each edge $uv \in D$ represents a traffic demand of value d_{uv} between its end nodes u and v. For each supply edge $e \in E$ one has to choose a capacity expansion y_e from a small set of possible choices with associated costs. This should be done so that the network (G, y) (with capacity vector y) can support the required traffic with total cost as low as possible.

The possible capacity choices on each edge gives rise to a discrete cost function, which can be modeled as follows. For each edge e let the incremental capacity steps be $m_e^t > 0$, for $t = 1, \ldots, T_e$ and let the incremental cost steps be $c_e^1, \ldots, c_e^{T_e} \geq 0$. The cost of installing a capacity y_e at edge e with $\sum_{t=0}^{s} m_e^t < y_e \leq \sum_{t=0}^{s+1} m_e^t$ is $\sum_{t=1}^{s+1} c_e^t$ for $s = 0, \ldots, T_e$. The jump in costs occuring for each capacity $\sum_{t=0}^{s} m_e^t$ may be due to e.g. the installation of a new cable. We let $m_e^{T_e} = \sum_{uv \in D} d_{uv}$ have a "very high" associated cost $c_e^{T_e}$; this means that all demands may be routed through any edge (but at a high cost). We model the cost function using a binary variable x_e^t for each incremental capacity step t on each supply edge e. For each e the variables $x_e^1, \ldots, x_e^{T_e}$ are required to be a sequence of ones followed by the sequence of zeros; this determines the capacity range. The index set of these (design) variables x_e^t is $I := \{ (e, t) \mid t = 1, \ldots, T_e, e \in E \}$, and $x \in \mathbf{R}^I$ is a **design vector** consisting of all these variables (with some ordering). For a design vector $x \in \mathbf{R}^I$ the corresponding cost is $c^T x$ and the **associated capacity vector** y is given by $y_e = \sum_{t=0}^{T_e} m_e^t x_e^t$ (where $x_e^0 := 1$).

We model the flow requirements as follows. Let $\bar{y}$ be the capacity

vector associated with some design vector x. The network $(G, \bar{y})$ is supposed to allow a multicommodity flow carrying all traffic (in which case $\bar{y}$ is called feasible). This requirement on $\bar{y}$ may be expressed in terms of linear inequalities as follows. For some given nonnegative vector $\mu \in \mathbf{R}^E$ and demand edge $f \in D$ let π_f^μ denote the shortest path length in G between the two end nodes of f with respect to edge lengths μ_e, $e \in E$. Then $\bar{y}$ is feasible if and only if

$$\sum_{e \in E} \mu_e \bar{y}_e \geq \sum_{f \in D} \pi_f^\mu d_f \quad \text{for all } \mu \geq 0. \tag{1}$$

This characterization of feasible capacities is known as the "Japanese theorem" (first stated in [8, 13]) and may be proved using linear programming duality (some more details are found in Section 2, see also [10]). We call the each inequality in (1) a **metric inequality**, see [16] for more comments on these inequalities.

In (1) we can restrict ourselves to a *finite* set of these inequalities; namely those defined by vectors (μ, π) in the set Π of extreme rays of the cone $\{\, \mu \in \mathbf{R}^E, \pi \in \mathbf{R}^D \mid \mu \geq 0,\ \pi_f = \pi_f^\mu \text{ for all } f \in D \,\}$. An important special case of the metric inequalities is obtained by choosing μ as the incidence vector of the **cut** $\delta_G(W)$ induced by a node set $W \neq \emptyset$, $W \neq V$ (when we assume that $G[W]$ and $G[V \setminus W]$ are connected). Then (1) reduces to the **cut inequality**

$$\bar{y}(\delta_G(W)) \geq d(\delta_H(W)). \tag{2}$$

This inequality assures that the total capacity of a cut is no smaller than all the demands across this cut. The other metric inequalities may be viewed as surplus conditions for more general structures than cuts in the graph.

Let $G = (V, E)$ and $H = (V, D)$ be as above. We model the network failures as follows. Consider a failing component $s \in V \cup E$. For a capacity vector $y \in \mathbf{R}^E$, the supply network $(G(s), y(s))$ is the network obtained by deleting s and setting $y(s)$ to zero for all deleted edges. The demand "network" $(H(s), d(s))$ is defined as (H, d) for $s \in E$ and $(H - s, d(s))$ for $s \in V$ (so $d(s)$ is zero for the deleted edges). When no network component is failing we set (artificially) $s = 0$, $(G(0), y(0)) = (G, y)$ and $(H(0), d(0)) = (H, d)$. The set of failure states is $S = V \cup E \cup \{0\}$ when the network is supposed to be survivable against node and edge failures, and $S = \{0\}$ when no survivbility is required.

The network is called **survivable** if for each $s \in S$ the supply network $(G(s), y(s))$ allows a multicommodity flow for the demand network $(H(s), d(s))$.

More complex survivability requirements could be treated similarly (see [5]).

The integer linear programming formulation of the MULTISUN problem with survivability constraints becomes now

$$\begin{array}{l} \min\ c^T x \\ \text{subject to} \\ \begin{array}{lll} \text{(i)} & 1 \geq x_e^1 \geq \cdots \geq x_e^{T_e} \geq 0 & \text{for all } e \in E; \\ \text{(ii)} & \sum_{e \in E(s)} \mu_e y_e \geq \sum_{f \in D(s)} \pi_f^\mu d_f & \text{for all } \pi \in \Pi(s),\ s \in S; \\ \text{(iii)} & y_e = \sum_{t=1}^{T_e} m_e^t x_e^t & \text{for all } e \in E; \\ \text{(iv)} & x_e^t \text{ integer} & \text{for all } (e,t) \in I. \end{array} \end{array} \tag{3}$$

Here, for each s, π_f^μ is the shortest-path length between the end nodes of demand edge f using length function μ and the set $\Pi(s)$ is the set of extreme rays of a certain cone (see the discussion above concerning metric inequalities).

A variation of this model is obtained by introducing additional **diversification constraints** on the flows in the normal state $s = 0$. This purpose of such constraints is to reduce the immediate loss of traffic when a failure occurs. For $0 < \lambda_{uv} \leq 1$ we say that a uv-flow z^{uv} of value d_{uv} in a network (G, y) is λ_{uv}**-diversified** if $z_e^{uv} \leq \lambda_{uv} d_{uv}$ for each $e \in E$ and $z^{uv}(\delta_G(w)) \leq 2\lambda_{uv} d_{uv}$ for each $w \in V - \{u, v\}$. This means that the uv-flow through any node or edge does not exceed $\lambda_{uv} d_{uv}$. (Here the edge requirement is only needed whenever $[u, v] \in E$; otherwise they are implied by the node requirements). The MULTISUN problem **with diversification** is the problem where we require each uv-flow to be λ_{uv}-diversified in state $s = 0$. We can model this problem by replacing the metric inequalities for $s = 0$ in (3) by the so-called **diversified metric inequalities**. Th eorigin of these inequalities is explained in Section 3.

2.2 Associated polytopes and improved formulations

We introduce the integral polytopes

$$\mathrm{MSUN}_S := \mathrm{conv}\{\, x \in \mathbf{R}^I \mid x \text{ satisfies (3)(i)–(iv)}\,\}. \tag{4}$$

The MULTISUN problem may be viewed as the LP problem

$$\min c^T x \quad \text{subject to } x \in \mathrm{MSUN}_S. \tag{5}$$

Note here the dependency on the failure state set S.

The polytope MSUN_S has a complicated polyhedral structure; see [16] for a study of some of its properties. We repeat a few results that are of interest for the cutting plane algorithm we use.

Under quite weak conditions MSUN_S is fulldimensional and all the ordering constraints (3)(i) define facets of MSUN_S (i.e., are nonredundant). The inequalities (3)(ii) do not define facets of MSUN_S except in very special cases. This indicates the need of stronger formulations than the "naive" LP relaxation given by (3), and, in fact, numerical results confirm this belief. We describe next how we obtain tighter LP formulations.

The **band inequalities** constitute a class of valid inequalities for MSUN_S that arise from a relaxation of MSUN_S, the so-called ICOV-polytope. Let $\sum_{e\in E}\sum_{t=1}^{T_e} g_e^t x_e^t \geq b$ be a metric inequality (3)(ii) or a diversified metric inequality, where $g_e^t := \mu_e m_e^t$. Because the highest capacity of each edge is "large", we may assume that $g_e^{T_e} \geq b$ for each e. Let $F := \{e \in E \mid g_e^1 > 0\}$, and consider the polytope $\mathrm{ICOV}(g,b) := \mathrm{conv}\{\, (x_e^t : t = 1,\ldots,T_e, e \in F) \mid \sum_{e\in F}\sum_{t=1}^{T_e} g_e^t x_e^t \geq b,$ $1 \geq x_e^1 \geq \cdots \geq x_e^{T_e} \geq 0$ for all $e \in F$, x integral$\,\}$. The polytope $\mathrm{ICOV}(g,b)$ can be viewed as a knapsack polytope with additional ordering constraints. Facial properties of related knapsack polytopes have been studied in [11, 17]. Any inequality that is valid for ICOV (g,b) is clearly also valid for MSUN_S, if the "missing" coefficients for indices (e,t) with $e \notin F$ are set to zero. For each $F \subseteq E$ we define the index set $I(F) := \{\,(e,t) \in I \mid t = 1,\ldots,T_e, e \in F\,\}$. For simplicity, we write $I(e)$ in stead of $I(\{e\})$. A **band** B of F is a subset of $I(F)$ containing exactly one element (e,t_e^B) in each $I(e)$, $e \in F$. Let $B^< = \{(e,t) \in I(F) \mid t < t_e^B\}$. A band B is called **valid** if $g(B^<) < b$

and in that case the **band inequality**

$$x(B) := \sum_{(e,t)\in B} x_e^t \geq 1, \tag{6}$$

is valid for ICOV(g, b).

Figure 1 illustrates a band inequality with $F = \{e_1, \ldots, e_4\}$ and $b = 4$. The x-axis denotes edges, and along the y-axis boxes of height g_e^t are stacked with g_e^0 lowest and $g_e^{T_e}$ highest. The (valid) band is the set of boxes marked with 1, and $g(B^<)$ equals the area below the band.

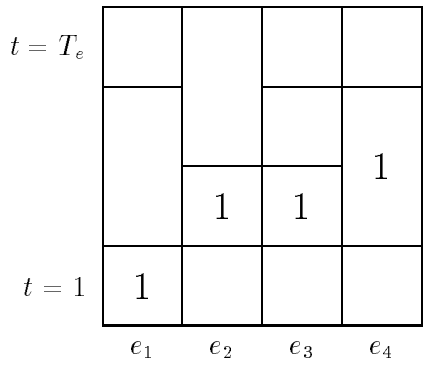

Figure 1: Band inequality

In [16] is is shown that if B is a band in F, where $|F| \geq 2$, then the band inequality $x(B) \geq 1$ defines a facet of ICOV(g, b) if and only if there is no valid band "above B". Since ICOV(g, b) is a relaxation of MSUN$_S$, for any nonempty S, each band inequality is also valid for MSUN$_S$, and under suitable additional conditions (depending on S) it will also define a facet of MSUN$_S$ for $S = \{0\}$.

The separation problem for band inequalities is NP-complete. In fact, this problem is equivalent to the NP-hard knapsack problem with ordering constraints. In Subsection 3.4 we discuss algorithms for solving this separation problem.

As mentioned, the band inequalities are valid for MSUN$_S$ for all choices of the failure state set S, although their strength may vary. Consider now the case when $E \subseteq S$, i.e., we include edge survivability.

Then the band inequalities are redundant, but we can often find a strengthened inequality as described next. Let $g^T x \geq b$ (where $b > 0$) be a metric inequality valid for the normal state $s = 0$, and let B be a band of F where $F = \{e \in E \mid g_e^1 > 0\}$. If $g(B^< \setminus I(e)) < b$ for all $e \in F$, then it can be shown that the **strengthened band inequality**

$$x(B) \geq 2 \tag{7}$$

is a valid inequality for MSUN_S, and it defines a facet of this polytope under rather weak conditions.

When a strengthened band inequality is derived from a cut inequality, it will, in many cases, define a facet of MSUN_S.

Finally, we remark that we have found other classes of facet defining inequalities that may be of interest. These include generalized band inequalities and partition inequalities arising from node partitions into three or more subsets. We have not included any of these inequalities in the algorithms reported here, but they may be of interest in further work.

3 Description of the algorithms

In this section we describe a cutting plane algorithm for solving the the MULTISUN problem based on the model given in (3). We also give algorithms for solving the associated separation problems and testing the feasibility of a given capacity vector $\bar{y}$. Finally, some simple primal heuristic methods are described.

3.1 The master problem

We use a cutting plane approach to the model (3). This means that we solve a sequence of sucessively stronger LP relaxations of (3), where each LP is obtained from the previous one by adding certain band inequalities that were violated by the previous optimal LP solution. In each iteration one determines whether the capacities obtained are feasible. One could view this approach as applying Benders' decomposition to a certain mixed integer linear programming model (see Subsection 3.2).

Master algorithm:

0. (Initialize) Find an initial relaxation P^0 of MSUN_S defined by selected band inequalities. Set iteration count $t := 0$.

1. (Master optimization) Solve the LP relaxation $\min \{c^T x \mid x \in P^t\}$ and obtain an optimal (vertex) solution x^t. Let $\bar{y}^t$ be the associated capacity vector.

2. (Master separation) Check if $\bar{y}^t$ satisfies all the metric inequalities (1). If it does, go to Step 3. If a violated cut inequality was found, use this to find violated band inequalities (see Subsection 3.4). Otherwise try a heuristic for finding violated band inequalities (using a pool of cuts). Let P^{t+1} be the polyhedron obtained by adding these band inequalities to P^t, set $t := t+1$ and return to Step 1. If no more band inequalities could be found, proceed to Step 3.

3. (Optimality check and heuristics) If x^t is integral, then x^t is optimal, and one terminates. Otherwise, let $z^{lo} = c^T x^t$ (lower bound) and use a primal heuristic (Subsection 3.5) for finding an upper bound z^{up} on the optimal value z^* in (3), and conclude that $z^{lo} \leq z^* \leq z^{up}$; terminate.

For solving the LP's in Step 1 we use the LP solver CPLEX, see [4]. The next subsections contain descriptions of algorithms used in Step 2 and 3.

3.2 Testing feasibility of multicommodity flow

We describe a method which determines whether a given capacity vector allows a multicommodity flow in all failure states $s \in S$, or equivalently whether $y \geq 0$ satisfies (3)(ii). We use techniques based on linear programming with row and constraint generation using a path formulation. An ordering of the set of operating states is selected and for each state $s \in S$ we solve a multicommodity feasibility problem with supply and demand graphs $G(s)$ and $D(s)$. We describe the algorithm for the case $s = 0$ only. The other problems ($s \neq 0$) are solved similarly. The optimal basic solution for state s is used as a starting solution for state $s+1$; this speeds up the algorithm considerably as consecutive problems tend to be very "near each other". To simplify the presentation we leave out the index s in matrices etc. below. In addition, we assume that $[u, v] \notin E$ for each demand edge $uv \in F$ (the general case is treated quite easily in our algorithm by adding suitable constraints). We also assume that the edges of positive capacity define a connected graph.

Let $f(uv)$ be a column vector with one element $f(uv,P)$ for each $[u,v]$-path P in G. Let $\mathcal{P}(uv)$ denote the set of all $[u,v]$-paths in the graph G. To solve the multicommodity feasibility problem for $s=0$ we set up the following linear programming model (**MF**). This is the well-known path formulation for multicommodity flow problems.

$$
\begin{array}{lrrcll}
\text{minimize } \alpha & & & & & \\
\text{subject to} & & & & & \\
\text{(i)} & \sum_{uv\in D} A(uv)f(uv) & -\bar{y}\alpha & \leq & \bar{y} & \\
\text{(ii)} & \mathbf{1}^T f(uv) & & = & d_{uv} & \text{for all } uv\in D; \\
\text{(iii)} & \sum_{P\in\mathcal{P}(u,v):w\in P} f(uv,P) & & \leq & \lambda_{uv}d_{uv} & \text{for all } uv\in D, \\
 & & & & & w\in V\setminus\{u,v\}; \\
\text{(iv)} & f(uv) & & \geq & 0 & \text{for all } uv\in D.
\end{array}
\tag{8}
$$

The **path variables** $f(uv,P)$, for each $P\in\mathcal{P}(uv)$ and $uv\in D$, express the flow on this path, and the **expansion variable** α represents an artificial capacity extension on each edge. The constraints (i) and (ii) assure that edge capacities are not exceeded and that all demands are satisfied. Finally, the **diversification constraints** (iii) assure that at most a given fraction λ_{uv} of demand uv is routed through the node $w\neq u,v$. The capacity $\bar{y}$ allows a diversified multicommodity flow if and only if the optimal value α^* in (MF) is nonpositive (so there is no need for an additional link capacity). Note that we may terminate the algorithm if we obtain a feasible solution of (MF) with $\alpha\leq 0$. Since $\bar{y}$ defines a connected graph, (MF) has a feasible solution. Similar path formulations are well known in the literature for "ordinary" multicommodity flow problems (without flow diversification) and originate from [6], see also [12]. For a recent fast combinatorial heuristic algorithm for the multicommodity flow problem, see [9].

For realistic problems the number of path variables and diversification constraints is normally very high, and it can be expected that in an optimal solution most variables are zero and few diversification constraints constraints active. We therefore solve (MF) by a row and column generation procedure.

In the column generation phase one solves shortest path problems, one for each demand; these weights are determined from the dual variables. Whenever a set of new path variables are added to the LP we also add diversification constraints for all the nodes in these paths.

This is done to avoid violation of these constraints for the next LP solution, and reduces the number of LP's to be solved. If the optimal solution α^* is positive, i.e., the capacity $\bar{y}$ is infeasible, then we obtain a violated inequality from the dual objective function. When the diversification constraints (8)(iii) are not present, this inequality is a metric inequality (3)(ii); otherwise it is a diversified metric inequality.

It may be needed to remove columns and/or constraints during the computations if the LP's become too large, but this is not done in the present implementation of these algorithms. (In particular, the number of diversification constraints may grow fast and should be controlled.) In our algorithm, the row generation is done "in advance"; we add all those diversification constraints that can possibly be violated by the next LP solution.

The algorithm is initialized by adding "promising" columns based on calculations of pairs of disjoint paths. To initialize the column generation in the various failure situations, one may use the traffic-carrying paths belonging to optimal routings in previously solved LPs.

Finally, when $\bar{y}$ does not define a connected graph, a cut inequality must be violated. This is determined without solving (MF) by searching components in the graph of edges with positive $\bar{y}_e$. In (MF) we remove constraints (ii) for edges e with $\bar{y}_e = 0$. When a violated metric inequality is found, one has to find the missing coefficients μ_e for the removed constraints. This entails shortest path computations for each missing edge. Instead, we only use those metric inequalities that define cut inequalities (2), because for those one can easily determine the missing coefficients.

3.3 The initial LP of the master problem

We decribe how to generate the first LP to be solved in the master problem. This is done heuristically by generating band inequalities from some "promising" cut inequalities that are likely to be violated by initial LP solutions. The purpose of the procedure is to (hopefully) reduce the number of calls to the time-consuming multicommodity feasibility routine (see Subsection 3.2).

The inequalities we use in the initial LP are the (strengthened) band inequalities (6) and (7). To determine a subset of these exponentially many inequalities, we use a dual ascent approach to the

following relaxation of min $\{c^T x \mid x \in MSUN_S\}$:

$$
\begin{array}{lrll}
(P) & \min K^T z & & \text{subject to} \\
 & H^T z & \geq 1 & \text{for certain band ineq. (6);} \\
 & H^T z & \geq 2 & \text{for certain strengthened band ineq. (7);} \\
 & z & \geq 0. &
\end{array}
$$

Here, $H^T z \geq 1$ ($H^T z \geq 2$) are band inequalities derived from cut inequalities (2) and K is the cost function, respectively. Let (D) denote the dual linear program to (P). A dual variable $\xi_H \geq 0$ is assigned to each band inequality $H^T z \geq b_H$ (where $b_H \in \{1, 2\}$) and the dual program becomes

$$
\begin{array}{lrll}
(D) & \max \sum_H b_H \xi_H & & \text{subject to} \\
 & \sum_H H_e^t \xi_H & \leq K_e^t & \text{for all } e \in E,\ t = 1, \ldots, T_e; \\
 & \xi_H & \geq 0. &
\end{array}
$$

The dual ascent method is a greedy algorithm for (D). It starts from the feasible dual solution $\xi := 0$ and increases certain variables ξ_H as much as possible without ξ_H becoming infeasible. The band inequalities H with positive ξ_H will then constitute the first LP, together with ordering constraints (3)(i).

Some more details are given next. The algorithm starts out with a set $\mathcal{F}$ of cuts being the one-node cuts $\delta_G(v)$ with incident demand edges. Thus the shores of the cuts in $\mathcal{F}$ are pairwise disjoint, and this property is maintained throughout the algorithm.

A dual ascent iteration

1. If $\mathcal{F}$ is empty, stop. Otherwise construct one (strengthened) band inequality $H^T z \geq 2$ or $H^T z \geq 1$ from each $F \in \mathcal{F}$.

2. Increase all ξ_H for the band inequalities constructed in Step 1 by a value α which is chosen largest possible without ξ becoming infeasible. Let f be an edge in the shore of some cut that defines a "tight" dual constraint.

3. Remove the (one or two) cuts containing f from $\mathcal{F}$. If there was only one such cut $\delta(W)$, and $i \in W$, then add the cut $\delta(W \cup \{j\})$ to $\mathcal{F}$. If there were two cuts containing f, with disjoint shores W and W', then add the cut $\delta(W \cup W')$ to $\mathcal{F}$, if some (strengthened) band inequality can be constructed from it.

3.4 Separation of band inequalities

We describe an algorithm for finding violated band inequalities in the master problem. As remarked before, this problem is hard, so we use heuristics for finding such violated inequalities.

Assume that a point $\bar{x} = (\bar{x}_e^t : \ e \in E \text{ and } t = 1, \ldots, T_e)$ is given, such that the vector $\bar{y}$ with components $\bar{y}_e := \sum_t m_e^t \bar{x}_e^t$ (for $e \in E$) violates some metric inequality $a^T y \geq b$ all of whose coefficients are integer. We describe a heuristic that derives a (possibly) violated band inequality (6) from the metric inequality.

Let $F := \{e \mid a_e > 0\}$, $g_e^t := a_e m_e^t$ for all $e \in F$ and $t = 1, \ldots, T_e$, and, finally, $G_e^t := \sum_{\tau=1}^{t} g_e^\tau$ for all $e \in F$ and $t = 1, \ldots, T_e$. We assume that all G_e^t and b are integer, and that

$$\begin{array}{l} \sum_{e \in F} G_e^1 < b \quad \text{and} \quad \sum_{e \in F} G_e^{T_e} \geq b \ ; \\ 0 < G_e^1 < \cdots < G_e^{T_e}; \\ \qquad \bar{x}_e^1 > \cdots > \bar{x}_e^{T_e} \geq 0 \end{array} \tag{9}$$

for all $e \in F$. If the monotonicity conditions on the $\bar{x}$ are not satisfied, say if $\bar{x}_e^t = \bar{x}_e^{t+1}$ for some e and t, one may remove variable x_e^t and renumber all following ones.

We are looking for 0/1 coefficients h_e^t ($e \in F$, $t = 0, \ldots, T_e$), such that the requirements (i)–(iii) in the definition of the band inequality are met. This can be formulated as the following integer LP:

$$\begin{array}{l} \min_h \sum_{e \in F} \sum_{t=1}^{T_e} \bar{x}_e^t h_e^t \\ \text{subject to} \\ \qquad \begin{array}{rcll} \sum_{e,t} G_e^t h_e^t & \leq & b-1; & \\ \sum_t h_e^t & = & 1 & \text{for all } e \in F; \\ h_e^t & \geq & 0 & \text{for all } e \in F, \text{ for all } t; \\ h_e^t & \text{integer} & & \text{for all } e, t. \end{array} \end{array} \tag{10}$$

Because the $\bar{x}_e^t$ decrease as t increases for fixed $e \in F$, the integer solution to (10) must satisfy the maximality requirement of (6).

The problem (10) for given G_e^t and $\bar{x}$ is NP-hard as the knapsack problem is a special case. So instead of solving the integer LP, we will solve its continuous relaxation. The LP-solution will contain at most two noninteger entries. By rounding, a "good" band inequality can be derived.

The LP-dual to the continuous relaxation of (10) is

$\max_{\alpha,\beta} \sum_{e\in F} \beta_e \; - (b-1)\,\alpha$
subject to

$$\begin{aligned} \beta_e \qquad - G_e^t\,\alpha \;&\leq\; \bar{x}_e^t \quad \text{for all } e \in F,\ t = 1,\ldots,T_e; \\ \alpha \;&\geq\; 0 \end{aligned}$$

For given α, the β_e can be computed as

$$\beta_e(\alpha) := \min_t \{\, \bar{x}_e^t + G_e^t\,\alpha \,\}$$

so the dual problem becomes

$$\max_{\alpha\geq 0} \phi(\alpha) \quad \text{where} \quad \phi(\alpha) = \sum_{e\in F} \beta_e(\alpha) - (b-1)\,\alpha \tag{11}$$

ϕ is a one-dimensional, concave, and piecewise-linear function, whose breakpoints are the breakpoints of the functions β_e. It has a finite maximum $\bar{\alpha} > 0$, because for large α, the $\beta_e(\alpha)$ equal $\bar{x}_e^1 + G_e^1\,\alpha$ for each e, and, by (9), $\sum_{e\in F} G_e^1 \leq b-1$. The value $\alpha = 0$ is not optimal, because $\sum G_e^{T_e} \geq b$, according to (9). We solve $\max\{\phi(\alpha) \mid \alpha \geq 0\}$ by a line-search procedure. From the optimal solution of (11) one can derive an optimal fractional solution to (10) and an integer (possibly non-optimal) solution by rounding.

3.5 Primal heuristics

The cutting plane algorithm often stops with a fractional solution $\bar{x}$, for which no more violated inequalities can be found. We describe here how to derive a feasible integer solution from a nonfeasible $\bar{x}$.

We implemented two methods. The first is called INCREASE. It increases fractional components, as it descends through a branch& bound tree. The other is called DECREASE, because it blows up $\bar{x}$ to a feasible but expensive solution and then greedily decreases its components. A more detailed description follows.

INCREASE scans $\bar{x}$ for fractional components. If there are none, then INCREASE stops unsuccessfully and one has to try another heuristic. Otherwise, let $\bar{x}_e^t$ be the largest fractional component among those with value < 1. Under these conditions, the index t is chosen as large as possible (if there were several possibilities). Then one transforms $1 \geq \bar{x}_e^1 \geq \bar{x}_e^2 \geq \ldots \geq \bar{x}_e^t$ into equations in the LP. Thereby x_e^t

is fixed to 1. Moreover, one fixes all components of $\bar{x}$ of value 1 to 1 for the rest of the heuristic. The new LP is solved with a new $\bar{x}$ as its solution. If $\bar{x}$ is still nonfeasible, one scans for fractional components, etc.

DECREASE first blows up $\bar{x}$ to a feasible solution x' as follows: it solves the feasibility test (8) for the normal state $s = 0$ with the capacity vector $\bar{y}$ (associated with $\bar{x}$). The expression $\sum_{uv \in D} A(uv)f(uv)$ in 8(i) defines a feasible capacity vector for normal state $s = 0$ whenever $\bar{y}$ was nonfeasible. In the same way one finds feasible capacity vectors for all failure states $s \in V \cup E$. By taking the component-wise maximum of all these vectors and then rounding each component to the next highest admissible capacity step, one finds a feasible integer solution x'. Now one decreases each component in x' as much as possible without violating feasibility. We have implemented several different ways of ordering the components, for instance based on smallest cost increase or smallest value of $\bar{x}_e^t - \bar{x}_e^{t-1}$. The number of feasibility test can be reduced by performing the described "blowing up" operation for each reduced x'. This operation does not increase any edge capacity (since x' is feasible) but may decrease more than one edge capacity, for example on induced paths.

Usually INCREASE is faster and produces better solutions. But sometimes DECREASE performs better. In our computational tests we have only used INCREASE.

4 Computational results

The algorithms have been implemented in C++ as a part of a network planning tool (called MULTISUN) used in Norwegian Telecom. An important part of this program is a graphical interface used to display and edit input networks and to show solutions with installed capacities. The computations were done on a DECstation 3100.

All the test runs reported below are from data supported to us by network planners. This means that these runs are of interest in the overall planning process. It should be remarked, however, that the final designs are typically decided by also taking into account other aspects (flexibility, budget etc.). Thus, the concept of an "optimal solution" should be interpreted correctly.

The test examples fall into four classes, each of these correpond to

a certain supply graph. The instances in a class are distinguished by different demands and survivability requirements. In particular, the diversification parameters λ have been varied (but kept the same for all demands in a given instance). We also vary a **reserve parameter** $r \in \{0, 1/2, 1\}$ which means that in case of node or edge failure the flow to be routed for each demand uv is rd_{uv}. Our main goal has been the case $r = 1$, but in the actual planning other values of r are also of importance.

The columns contain the following information:

- λ: diversification parameter
- r: reserve parameter
- Lbd: best lower bound on the optimal value
- Ubd: best upper bound on the optimal value (normally from INCREASE heuristics)
- Gap: (Ubd - Lbd)/Lbd in percent
- Time: CPU time in minutes.seconds
- LP: number of master LP problems
- Band: number of band inequalities in final master LP

The test class A consists of 12 instances with the same supply graph having 27 nodes and 51 edges, see Table 1. The cost function has six steps (0, 63, 252, 1008, 5040 and 11088) and has a clear "concave" structure (the actual cost naturally involves the distance between the end nodes).

In Table 2 one finds test class B with the supply graph having 118 nodes and 134 edges. This graph is very sparse, which is the case for many interesting applications. The cost function varies from one edge to another, altogether ten different functions are used. They all have 5 steps and a certain free capacity which ranges from 0 to 110.

The supply graph of test class C has 37 nodes and 44 edges. Six quite different cost functions are used. For instance, one cost function has steps for capacities 5, 63, 252 and 11088 (which is large compared to all demands), while another has steps for 0 (i.e., no free capacity), 63, 252 and 1004.

The last test class D is with a supply graph having 45 nodes and 53 edges with cost functions roughly as for class C.

Our experiences may be summarized as follows. The gap is generally very low, and often less than 1 percent. However, this only applies

200	100	50	20	λ	r	Lbd	Ubd	Gap	Time	LP	Band
	19			0.5	1.0	282.7	287.7	*1.7*	2.54	8	440
	19			1.0	1.0	282.7	287.7	*1.7*	7.14	10	454
	10	9		0.5	1.0	279.3	305.1	*9.3*	5.55	13	455
	10	9		1.0	1.0	279.4	305.1	*9.2*	14.47	13	462
7	6	6		0.5	1.0	295.8	321.4	*8.7*	7.02	18	523
7	6	6		1.0	1.0	295.9	314.2	*6.2*	19.14	21	614
	5		14	0.5	0.5	236.3	237.3	*0.4*	1.42	9	207
	5		14	0.5	1.0	266.4	267.9	*0.6*	2.29	5	215
	5		14	1.0	1.0	266.4	267.9	*0.6*	1.26	5	216
	14		5	0.5	0.5	274.7	282.7	*2.9*	13.49	12	628
	14		5	0.5	1.0	279.4	302.1	*8.1*	4.54	7	347
	14		5	1.0	1.0	279.4	302.1	*8.1*	14.43	7	355

Table 1: Test set A

5	2	1	λ	r	Lbd	Ubd	Gap	Time	LP	Band
	113		0.5	1.0	56.7	56.7	*0.0*	11.08	22	400
	113		1.0	1.0	56.7	56.7	*0.0*	11.01	21	445
		113	0.5	1.0	54.1	54.1	*0.0*	0.39	5	109
		113	1.0	1.0	54.1	54.1	*0.0*	0.28	5	109
	56	57	0.5	1.0	55.9	55.9	*0.0*	4.00	18	225
	56	57	1.0	1.0	55.9	55.9	*0.0*	3.18	18	225
	56	57	0.5	0.5	53.9	54.0	*0.2*	3.12	3	103
5	20	5	0.5	1.0	38.1	38.1	*0.0*	0.16	12	238
5	20	5	1.0	1.0	38.1	38.1	*0.0*	0.17	12	238
5	20	5	1.0	0.5	37.7	37.7	*0.0*	0.07	9	197
20	5	5	1.0	0.5	37.7	37.7	*0.0*	0.07	8	198
20	5	5	1.0	1.0	39.8	39.9	*0.2*	1.36	21	412
20	5	5	0.5	1.0	39.8	40.0	*0.4*	1.46	21	396

Table 2: Test set B

10	2	1	λ	r	Lbd	Ubd	Gap	Time	LP	Band
		25	1.0	0.0	12.1	17.9	*48.1*	0.14	14	85
	25		1.0	1.0	37.1	37.1	*0.0*	0.03	5	86
	25		0.5	0.5	37.1	32.8	*13.1*	0.13	6	89
	25		0.5	1.0	37.1	37.1	*0.0*	0.03	5	86
3	11	11	0.5	0.5	37.1	37.1	*0.0*	0.04	6	101
3	3	8	0.5	1.0	28.2	28.2	*0.0*	0.02	5	89
5	5	4	0.5	1.0	28.2	28.2	*0.0*	0.01	7	93
5	5	4	1.0	1.0	28.2	28.2	*0.0*	0.01	7	93
5	9		0.5	1.0	28.2	28.2	*0.0*	0.01	7	95
5	9		1.0	0.5	28.2	28.2	*0.0*	0.02	12	122
10	4		1.0	0.5	28.2	28.2	*0.0*	0.03	8	123
10	4		0.5	0.5	28.2	28.2	*0.0*	0.03	8	123

Table 3: Test set C

30	20	10	λ	r	Lbd	Ubd	Gap	Time	LP	Band
		7	1.0	1.0	33.0	33.7	*2.1*	0.06	15	183
		7	0.5	1.0	33.0	33.7	*2.1*	0.06	15	183
		7	0.5	0.5	32.3	32.3	*0.0*	0.01	10	150
		7	1.0	0.0	14.3	17.5	*22.6*	0.04	6	52
	4	3	1.0	0.5	32.3	32.3	*0.0*	0.01	10	150
	4	3	0.5	0.5	32.3	32.3	*0.0*	0.01	10	150
2		3	0.5	1.0	33.0	33.7	*2.1*	0.05	12	155
2		3	0.5	0.5	32.3	32.3	*0.0*	0.01	8	115
2		3	1.0	0.5	32.3	32.3	*0.0*	0.01	8	115
3			1.0	0.5	27.3	28.0	*2.6*	0.02	11	143
3			0.5	0.5	27.3	28.0	*2.6*	0.02	11	143
3			0.5	1.0	27.3	28.0	*2.6*	0.03	11	143

Table 4: Test set D

to the survivability case where $r = 1$. For instance, as seen in Table C, the gap may become very large whenever $r = 0$, although such problems were not the main goal of this work. This phenomenon may be explained by the fact that our cutting plane algorithm only adds band inequalities derived from cuts. For connectivity design problems, see [15], it is known that cut inequalities give strong relaxations for 2-connectivity problems, but not for 1-connectivity (Steiner) problems. For problems with "low" r it therefore seems that adding partition inequalities as described in [16] would reduce the gap.

We also note that the number of LP's solved and the number of band inequalities are "under control". The computation time varies and most of it is spent in our multicommodity flow routine. It is clear that for solving larger problem one would benefit from developing faster approximation algorithms for multicommodity flows.

5 Conclusions

We have developed a model MULTISUN for the design of survivable networks allowing multicommodity flows. The survivability assures that the network has enough capacity to perform rerouting in case of a single node or edge failure. Furthermore flows are diversified in the nonfailure case. The cost function is a step function. Based on a strengthened formulation using band inequalities derived from cut inequalities we have described a cutting plane algorithm for MULTISUN. The computational results show that the real world planning problems at hand were solved to near-optimal solutions. The instances were fairly large although the supply graphs were sparse.

Further work could be directed towards the similar problem without node/edge survivability. There the development of separation heuristics for the class of partition inequalities (see [16]) is of interest. Another interesting area is to find better heuristics for the MULTISUN problem and approximation algorithms for the multicommodity flow feasibility problem.

Acknowledgements. The authors would like to thank Rune H. Johansen for developing the user interface and some basic algorithms in MULTISUN, as well as Per Johan Brun for performing numerical test runs.

References

[1] A. Balakrishnan and S. C. Graves, A composite algorithm for a concave-cost network flow problem. *Networks*, 19:175–202, 1989.

[2] C. Berge. *Graphs and Hypergraphs*. North-Holland, Amsterdam, 1973.

[3] D. Bienstock, S. Chopra, O. Günlük and C.Y. Tsai, Minimum cost capacity installation for multicommodity network flows. Draft. Columbia University, New York, January 1995.

[4] *Using the CPLEX callable library and CPLEX mixed integer library.* CPLEX Optimization, Inc., 1993.

[5] G. Dahl and M. Stoer, MULTISUN — mathematical model and algorithms. Technical Report TF R 46/92, Norwegian Telecom Research, Kjeller, Norway, 1992.

[6] L.R. Ford Jr. and D. R. Fulkerson, A Suggested Computation for Maximal Multicommodity Network Flows. Management Science, 5, 1968 (97–101).

[7] B. Gavish, P. Trudeau, M. Dror, M. Gendreau, and L. Mason, Fiberoptic circuit network design under reliability constraints. *IEEE Journal on Selected Areas in Communications*, 7(8):1181–1187, 1989.

[8] 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.

[9] T. Leong, P. Shor and C. Stein, Implementation of a Combinatorial Multicommodity Flow Algorithm. "DIMACS Implementation Challenge Workshop, Algorithms for Network Flow and Matching, DIMACS Technical Report 92-4", ed. D. S. Johnson and C. C. McGeogh, Jan. 1992 (202–224).

[10] M. V. Lomonosov. Combinatorial approaches to multiflow problems, *Discrete Applied Mathematics*, 11:1–93, 1985.

[11] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, Chapter: The Multiple-Choice Knapsack Problem, pages 77–80. Wiley, Chichester, 1990.

[12] 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.

[13] K. Onaga and O. Kakusho, On feasibility conditions of multicommodity flows in networks. *Transactions on Circuit Theory*, CT-18(4):425–429, 1971.

[14] A. Schrijver, *Theory of Linear and Integer Programming*. Wiley, Chichester, 1986.

[15] M. Stoer, *Design of Survivable Networks*, volume 1531 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1992.

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

[17] L. A. Wolsey, Valid inequalities for 0-1 knapsacks and MIPs with generalised upper bound constraints. *Discrete Applied Mathematics*, 29:251–261, 1990.