Telenor

Forskningsnotat/ Scientific Document

FoU N 78/95

SC 95-41

Geir Dahl, Alexander Martin, Mechthild Stoer

Routing through virtual paths in layered telecommunication networks

Telenor FoU

Telenor

Telenor Forskning og Utvikling
Instituttveien 23, P.O.Box 83, N-2007 Kjeller, Norway
Tel: +47 63 84 84 00 - Fax: +47 63 81 98 10

Telenor

Tittel

Routing through virtual paths in layered telecommunication networks

FoU-Notat
N78/95
ISBN
ISSN 0803-5652
Project no TFN9506A
Program
Security gr. Open
No. of pages 25
Date 95.12.07

Author(s)

Geir Dahl, Alexander Martin, Mechthild Stoer

Subject headings

Logiske veier, ruting, optimering

Abstract

We study a network configuration problem in telecommunications where one wants to set up logical paths in a capacitated network to accommodate given point-to-point traffic demand. The problem is formulated as an integer linear programming model where 0-1 variables represent different paths. An associated integral polytope is studied and different classes of facets are described. These results are used in a cutting plane algorithm. Computational results for some realistic problems are reported.

Title (English)

Routing through virtual paths in layered telecommunication networks

Routing through virtual paths in layered telecommunication networks [1]

Geir Dahl [2]
Alexander Martin [3]
Mechthild Stoer [4]

Abstract

We study a network configuration problem in telecommunications where one wants to set up paths in a capacitated network to accommodate given point-to-point traffic demand. The problem is formulated as an integer linear programming model where 0-1 variables represent different paths. An associated integral polytope is studied and different classes of facets are described. These results are used in a cutting plane algorithm. Computational results for some realistic problems are reported.

1 Introduction

A major trend in telecommunications is increased flexibility in terms of network configuration and resource allocation. In particular communication paths in networks may be set up on a temporary basis and controlled by software in order to meet changing demands due to, e.g., data communications or video applications. Such paths (often called virtual paths) have the attractive feature of low processing time in the intermediate nodes. An important problem area concerns the management of these capacitated paths, and in this paper we are concerned with such a problem in a two-layered network.

[1] submitted to *Operations Research*.

[2] University of Oslo, P.O.Box 1080, Blindern, N-0316 Oslo, Norway. Email: geird@ifi.uio.no.

[3] Konrad-Zuse-Zentrum für Informationstechnik, Heilbronner Str. 10, D-10711 Berlin, Germany. Email: martin@zib-berlin.de.

[4] Telenor Research and Development, P.O.Box 83, N-2007 Kjeller, Norway. Email: stoer@nta.no.

The model we study is as follows: One has given a set of point-to-point traffic demands that need to be routed in a so-called pipe-network. Each edge in this network is called an express pipe. It has a fixed, uniform capacity measured in the same units as the traffic demands. Each express pipe corresponds to a path in an underlying physical transmission network. When an express pipe is established, it uses resources in the transmission network, say, a fiber pair in a fiber cable. For each edge in the transmission network, one has therefore an upper bound on the number of pipes that can go through it. The problem is now to select some of the given express pipes such that the traffic can be routed upon them, taking into account express pipe capacity and physical link capacity. Costs are associated with the establishment of express pipes and with the routing. When we use the term "routing", we don't mean dynamic routing at call setup time. We focus rather on the setup of the express pipes which accommodate forecasted traffic and are not changed every few minutes. We also assume that the set of pipes to choose from, is given beforehand. Pipes are not generated dynamically in the course of the algorithm.

One motivation for studying the routing and path-packing model comes from routing and grouping in the PDH or SDH bandwidth hierarchy. There traffic given in 2 Mbit/sec is switched onto systems of different fixed bandwidths. A model involving several levels of networks and an LP-based solution method is described in [9].

Another application may be in ATM-networks. There traffic corresponds to virtual circuits, which can be packed into virtual paths (our express pipes). Our model should, however, be refined to capture this case. Especially do virtual paths take many bandwidths (not just one as in our model) in the physical network, the virtual paths don't "eat" capacity of the physical network in the form of fiber but in the form of bandwidth, and our cost function does not exactly model the gains (less call control in intermediate nodes) versus the disadvantages (splitting of bandwidth) of setting up virtual paths.

[14] and [15] describe integer programming algorithms for routing (unsplittable) demands in a capacitated network such as to maximize revenue and route as many demands as possible. This is the bandwidth packing problem. Our model is distinguished from theirs in that it involves the intermediate pipe layer, and the demand routing is modeled with flow variables instead of path variables.

This paper is organized as follows. In Section 2 the integer linear pro-

gramming model for the mentioned problem is presented. The body of this work is a polyhedral study which is found in Section 3. Various classes of facet defining inequalities are introduced. These inequalities are used to find stronger relaxations of the integer program, and in Section 4 we describe a cutting plane algorithm using such relaxations. Separation heuristics and primal heuristics are also discussed and computational results for some realistic problems are reported.

We use fairly standard notation from graph theory and polyhedral theory, see [4] and [18], respectively. However, 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. The **cut** $\delta_G(W)$ induced by a subset W of V in the graph G is the set of edges with one endnode 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 uv-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 uv-paths. When $a^T x \leq \alpha$ is a valid inequality for a polyhedron P we call each point $x_0 \in P$ with $a^T x_0 = \alpha$ a *root* (of the inequality $a^T x \leq \alpha$).

2 Mathematical model

In this section we give a mathematical formulation of the problem, describe it as an integer linear programming model and introduce an associated polytope corresponding to the feasible solutions. Some basic properties of these polytopes are discussed.

The physical network of interest is modeled as an undirected graph $N = (V, L)$ with node set V corresponding to switching nodes and edge set $l \in L$ corresponding to transmission lines (fiber cables). We call N the *physical graph* and its edges *physical edges (links)*. The traffic demands are modeled by the *demand graph* $D = (V, K)$ where each *demand edge* $[u^k, v^k] \in K$ represents a traffic demand between the endnodes u^k and v^k of size d^k. (Typically, there are several isolated nodes in the demand graph). The final in-

gredience of our model is the *pipe graph* $G = (V, E)$ where each *pipe (edge)* $e = [u, v] \in E$ corresponds to a uv-path in the physical graph N. A pipe may then represent a transmission path in the telecommunication network (possibly set up up for a limited time period) on which different traffic may be routed. Note that G may contain many parallel edges. One may view the whole network architecture as two-level hierarchical. Sometimes more than two levels are of interest, but we do not treat this here.

The model also incorporates capacities in the following way. Each demand should be routed in the pipe graph, i.e., each demand $k = [u, v]$ uses some uv-path $e_1, \ldots, e_t$ of pipe edges in G. We assume that the capacity of each pipe $e \in E$ is $B > 0$ meaning that the total demand that may be routed on each pipe may not exceed B. Furthermore, the number of selected pipes (in a feasible solution) containing a physical link $l \in L$ must not exceed the capacity c_l (we assume throughout that $c_l \geq 1$). This may, e.g., correspond to the situation where each pipe is allocated to an individual fiber on the fiber cable $l \in L$. Thus we have capacity constraints in both levels of the network architecture, both for "embedding" demands (connections) in the pipe graph, and for embedding pipes in the physical network.

The problem of interest is to select pipes that are to be used and to determine on which path of the selected pipe set each of the demands should be routed. The cost function is the sum of the costs ν_e for selecting a pipe e and the costs ω_e^k for routing a demand k through pipe e. This problem of finding a minimum cost pipe selection and routing is called the *pipe selection and routing problem* PIPE.

We model mathematically the PIPE problem as the following integer linear program

$$\min \sum_{e \in E} \gamma_e y_e + \sum_{k \in K} \sum_{e \in E} \omega_e^k x_e^k$$

subject to

$$\begin{array}{lll} \text{(i)} & x^k(\delta_G(W)) \geq 1 & \text{for all } W \subset V \text{ with } u^k \in W,\ v^k \notin W,\ k \in K; \\ \text{(ii)} & \sum_{k \in K} d^k x_e^k \leq B y_e & \text{for all } e \in E; \\ \text{(iii)} & \sum_{e : l \in e} y_e \leq c_l & \text{for all } l \in L; \\ \text{(iv)} & 0 \leq x_e^k \leq 1, 0 \leq y_e \leq 1 & \text{for all } k \in K,\ e \in E; \\ \text{(v)} & x_e^k,\ y_e \text{ are integer} & \text{for all } k \in K,\ e \in E. \end{array} \qquad (1)$$

The 0-1 variable y_e indicated whether pipe $e \in E$ is selected, and the variable x_e^k indicates if demand $k \in K$ uses (is routed on) pipe $e \in E$. Constraints (i) assures that x^k is the incidence vector of a pipe set containing a $u^k v^k$-path for each $[u^k, v^k] \in K$. This is due to Menger's connectivity theorem (see [12] or [2]). Constraints (ii) and (iii) reflect the capacity constraints in the pipe graph and the physical graph, respectively.

Remark. We shall assume throughout that the demand set K is partitioned into two subsets K_1 and K_2 such that $d^k = 1$ for $k \in K_1$ and $d^k = B$ for $k \in K_2$. (When $B = 1$, we let $K_1 = \emptyset$ and $K_2 = K$.) This is of interest in the applications we consider. Furthermore, in our implementation and numerical experiments we have restricted the attention to the parameter choice $\gamma_e = \gamma$ for all $e \in E$ and $\omega_e^k = d^k \omega_e$.

The PIPE problem (1) can be shown to be *NP*-hard as it contains the path packing problem (see [8]) as a special case. This problem is to decide if a given graph contains edge-disjoint paths each connecting a given pair of nodes. Thus, finding a theoretically efficient algorithm for PIPE is (probably) impossible. However, experience from other related problems ([7], [11], [15]) indicate that cutting plane algorithms may perform very well on practical problem instances. Our goal is to find strong relaxations for the integer program and use these to develop a cutting plane algorithm for PIPE.

We introduce a family of integer polytopes associated with the model in (1):

$$P_{\mathcal{S}} := \text{conv}\{\, (x,y) \in \mathbf{R}^{K \times E} \times \mathbf{R}^E \mid x, y \text{ satisfies (1) (i)–(v)} \,\}, \qquad (2)$$

where $\mathcal{S} = (N, G, D, B, d, c)$ specifies the instance and $c = (c_l : l \in L)$ and $d = (d^k : k \in K)$ are the capacity and demand vector, respectively. The PIPE problem may be viewed as the LP problem

$$\min\{f(x,y) \mid (x,y) \in P_{\mathcal{S}}\}, \qquad (3)$$

where $f(x,y)$ is the linear objective function in (1). In order to solve (1), or produce good lower bounds, one needs to find a "sufficiently" good approximation to a linear system of inequalities with solution set $P_{\mathcal{S}}$. The polytope $P_{\mathcal{S}}$ has a complicated polyhedral structure, and an analysis of some of its properties is made in the next section.

3 Polyhedral properties

The goal of this section is to establish a number of properties of the polytope P_S. We study the dimension of P_S and additional classes of inequalities that define facets of this polytope.

The problem of deciding whether P_S is nonempty (i.e., finding a feasible solution in (1)) is *NP*-complete. This follows from the fact that the special case of deciding the existence of edge-disjoint paths between specified terminals is *NP*-complete, see [8]. However, a criterion for fulldimensionality may be stated as follows.

Proposition 4 *P_S is fulldimensional if the PIPE instance $S(e) = (N, G \setminus \{e\}, D, B, d, c')$ is feasible for each $e \in E$, where $c'_l = c_l$ for all $l \notin e$ and $c'_l = c_l - 1$ for all $l \in e$.*

Proof. Assume that $S(e)$ has a feasible solution for each $e \in E$. Also assume that P_S is contained in the hyperplane defined by the linear equation

$$\sum_{e \in E} a_e y_e + \sum_{k \in K} \sum_{e \in E} b_e^k x_e^k = \alpha. \tag{5}$$

Let $e \in E$. By assumption there is a feasible solution (x, y) in (1) with $y_e = 0$ and with capacity function c'. Define $y' \in \mathbf{R}^E$ by $y'_f = y_f$ for $f \neq e$ and $y'_e = 1$. For each $l \in e$ we then have $\sum_{e': l \in e'} y'_{e'} \leq c'_l + 1 = c_l$, and it follows that (x, y') is feasible. Thus both (x, y) and (x, y') satisfy (5) and this implies that $a_e = 0$. As e was arbitrary, we get $a = 0$. Furthermore, let x' be obtained from x by setting $(x')_e^k = 1$ for some k. Then both (x, y') and (x', y') are feasible in (1) and therefore satisfy (5). This leads to $b_e^k = 0$ for all $k \in K$ and $e \in E$. Thus $a = 0$ and $b = 0$, which contradicts that the inequality in (5) defines a hyperplane. Therefore P_S is fulldimensional as claimed. □

We remark that in the case when $L = E$, a necessary condition for P_S to be fulldimensional is that $S(e) = (N, G \setminus \{e\}, D, B, d, c')$ is feasible for all $e \in E$.

All the inequalities in (1) define facets of P_S whenever the pipe graph is "dense" enough. We do not go into these details, but concentrate in the following on finding strengthened formulations. Several classes of new facet defining inequalities are introduced.

Knapsack inequalities

Each inequality in (1) (ii) may be viewed as a knapsack inequality. In fact, using the linear transformation $T_e(y) = z$ where $z_e = 1 - y_e$ for each $e \in E$ we get the knapsack inequality

$$\sum_{k \in K_1} x_e^k + B \sum_{k \in K_2} x_e^k + Bz_e \leq B. \tag{6}$$

Each valid inequality for the knapsack polytope defined by (6) is also valid for P_S when setting $z_e = 1 - y_e$.

For a study of different properties of knapsack polytopes, see [13] and the references cited there. Based on the knapsack inequality we obtain the following class of *cover inequalities* that are valid for P_S for each pipe edge $e \in E$:

$$x_e^{k_1} + \textstyle\sum_{k \in K_2} x_e^k \leq y_e \quad \text{for all } k_1 \in K_1. \tag{7}$$

A combinatorial interpretation of such an inequality is that if more than one demand is routed on e, then all these demands are K_1-demands. In certain special situations a complete linear description of knapsack polytopes has been found, see [19]. It follows from the results of [19] that a complete linear description of the knapsack polytope defined by (6) is given by the inequalities (6), (7) and simple bounds.

Note that if $|K_1| \leq B$, then the knapsack inequality (1) (ii) is dominated by the sum of cover inequalities. Under certain known conditions the cover inequalities define facets of the knapsack polytope (see [13] for a general discussion). With suitable additional assumptions on the PIPE instance $\mathcal{S}$, the cover inequalities also define facets of P_S.

Strengthened cut inequalities

Consider a cut $\delta_G(W)$ in the pipe graph, where W and $V \setminus W$ is nonempty. Let K' be the demands in $K_1 \cap \delta_D(W)$. Then

$$y(\delta_G(W)) - \sum_{k \in K_2} x^k(\delta_G(W)) \geq \lceil |K'|/B \rceil \tag{8}$$

is a valid inequality. To see this we add the inequalities

- $x^k(\delta_G(W)) \geq 1$ for all $k \in K'$

- $By_e - \sum_{k \in K_1} x_e^k - B \sum_{k \in K_2} x_e^k \geq 0$ for all $e \in \delta_G(W)$

and divide the result by B. We can round the coefficients of the left-hand side of this new inequality by adding an appropriate amount of $x_e^k \geq 0$, and round the right-hand side. The resulting valid inequality is (8) which we call the **strengthened cut inequality**. These inequalities are also nonredundant under reasonable conditions. To avoid technicalities, we will show this for highly-connected graphs G and N.

Lemma 9 *The strengthened cut inequality (8) defines a facet of P_S if the following conditions are satisfied.*

(i) $|K'|/B$ *is not an integer.*

(ii) There are at least max $\{\lceil d(\delta_D(w))/B \rceil + 1 \mid w \in \{u, v\}\}$ *parallel uv-pipes between any* $u, v \in V$, $u \neq v$.

(iii) $L = E$.

Proof. First, P_S is full-dimensional, because the conditions of Proposition 4 are satisfied. Consider a facet defining inequality

$$\sum_{e \in E} a_e y_e + \sum_{k \in K} \sum_{e \in E} a_e^k x_e^k \geq \alpha \tag{10}$$

such that each root of (8) satisfies (10) with equality. We will show that the coefficients of this inequality are as in the strengthened cut inequality.

For any set $F \subseteq \delta_G(W)$ of cardinality $t := \lceil (1/B) d(\delta_D(W)) \rceil$ there exists a feasible root solution in which $y_e = 1$ for $e \in F$ and $y_e = 0$ for $e \in \delta_G(W) \backslash F$, and in which all demands in $\delta_D(W)$ use at most three pipes, and all other demands use at most one pipe. This is due to assumptions (ii) and (iii). Since there is at least one pipe in each shore of the cut not used by such a solution, one can prove that a_e and $a_e^k = 0$ for all $e \in E[W]$ and all k. Because of condition (i) one can also find sufficiently many routings of small demands to prove $a_e^k = 0$ for $e \in \delta_G(W)$. Now compare a root solution using $F \subseteq \delta_G(W)$ with a root solution using $F - \{e\} + \{f\}$ for arbitrary edges $e \in F$ and $f \in \delta_G(W) \setminus F$. The routings in $F - \{e\}$ are supposed to be the same in both solutions, and e is supposed to carry only demands of $K' = K_1 \cap \delta_D(W)$. Note that, because of condition (i), K' is not empty! The comparison of the

two solutions proves that $a_e = a_f$. This is true for all $e, f \in \delta_G(W)$. Now compare a root solution using $F \subseteq \delta_G(W)$ with the root solution in which an arbitrary edge $e \in \delta_G(W) \setminus F$ is added to F, then y_e and some x_e^k for $k \in K_2$ is set to 1, and all other variables stay the same. This proves that $a_e^k = -a_e$. Since e, k, and F were arbitrary, (10) has the same coefficients as the strengthened cut inequality 8, hence it defines a facet. □

Note that condition (i) is also necessary for (8) to define a facet.

Remark. The strengthened cut inequalities may be generalized in the spirit of the "flow-cutset inequalities" introduced in [3]. Let F be a subset of $\delta_G(W)$. In the validity proof above add the inequalities $x^k(\delta_G(W))$ as before, but now add the knapsack inequalities only for $e \in F$. The resulting *flow-cutset inequality* is

$$y(F) + \sum_{k \in K'} x^k(\delta_G(W) \setminus F) - \sum_{k \in K_2} x^k(F) \geq \lceil |K'|/B \rceil.$$

Hypomatchable inequalities

We introduce and study a large class of inequalities called hypomatchable inequalities.

Consider an instance $\mathcal{S}$ of PIPE with $B \geq 2$. Choose an odd number of nodes $V' = \{v_1, v_2, \ldots, v_n\} \subseteq V$, and demands k_1, k_2, ..., k_n in K_1 (not necessarily distinct) such that demand k_i is incident to v_i. Lastly, choose a set $F \subseteq E[V']$ with the property that if $k_i = k_j$ then $[v_i, v_j]$ is not in F. Denote by K' the set of chosen demands with only one endpoint in V', and denote by K'' the set of chosen demands with two endpoints in V'. Let F' be the set F together with all edges $[v_i, v_j]$ with $k_i = k_j$.

Consider the inequality

$$y(F) - \sum_{k \in K_2} x^k(F) + \sum_{i=1}^{n} x^{k_i}(\delta_G(v_i) \setminus F') + \sum_{k \in K''} \sum_{\substack{e \in E: \\ e \text{ connects } u^k, v^k}} x_e^k \geq \lceil n/2 \rceil \quad (11)$$

which we call a *hypomatchable inequality*, because, as we shall see later, the inequality has a good chance to be facet-defining when (V', F') defines a hypomatchable graph.

Lemma 12 *The hypomatchable inequality (11) is valid for $P_{\mathcal{S}}$.*

Proof. Add the valid degree and cover inequalities

- $x^{k_i}(\delta_G(v_i)) \geq 1$ for $i = 1, \ldots, n$ and
- $y_e - x_e^{k_i} - \sum_{k \in K_2} x_e^k \geq 0$ and
  $y_e - x_e^{k_j} - \sum_{k \in K_2} x_e^k \geq 0$ for each $e = [v_i, v_j] \in F$.

Divide both sides by two, and round up all coefficients on the left-hand side by adding the corresponding nonnegativity constraints $\frac{1}{2}x_e^k \geq 0$. Since the left-hand side takes integer values for all $(x, y) \in P_S$, one can round up the right-hand side to get a valid inequality, namely (11). □

As an illustration, consider a three-node example with nodes v_1, v_2, v_3 and parallel pipes e_i and e_i' both with endnodes v_i and v_{i+1} for $i = 1, 2, 3$ (we identify v_4 and v_1). We also let $L = E$, $B = 4$. Demand k is parallel to e_k and $d^k = 1$ for $k = 1, 2, 3$. Let $F := \{e_1, e_2, e_3\}$ and define the fractional solution $(\bar{x}, \bar{y})$ by $\bar{x}_e^k = \bar{y}_e = 1/2$ if $e \in F$ and $\bar{x}_e^k = \bar{y}_e = 0$ for $e \in E \setminus F$. This solution corresponds to the nonintegral routing of each demand by splitting the flow equally along the two paths between each pair of nodes on the triangle. One can verify that $(\bar{x}, \bar{y})$ satisfies all the linear inequalities in (1) as well as the knapsack and cover inequalities (6), (7), and the strengthened cut inequalities (8). However, $(\bar{x}, \bar{y})$ violates the hypomatchable inequality $y(F) + \sum_{i=1}^{3} x^{k_i}(\delta_G(v_i) \setminus F) \geq 2$.

We discuss conditions under which a hypomatchable inequality is nonredundant. We introduce some convenient terminology. For a graph $H = (V, F)$ with an odd number of nodes, we call $M \subseteq F$ a *supermatching* if all nodes except one are incident to exactly one edge of M, and the last node is incident to two edges of M. A supermatching of H has $(|V| + 1)/2$ edges.

Consider a root (x, y) of a hypomatchable inequality (11), i.e., (x, y) is a feasible solution of (1) that satisfies (11) with equality. Let $M := \{[v_i, v_j] \in F' \mid x_e^{k_i} = 1 \text{ or } x_e^{k_j} = 1\}$. It can be seen that there is at most one isolated node in (V', M), and that M is either a supermatching with $\lceil n/2 \rceil$ or a matching with $\lfloor n/2 \rfloor$ edges.

A graph H is *hypomatchable* (see [10]) if $H \setminus \{v\}$ contains a perfect matching for each $v \in V[H]$. Examples of hypomatchable graphs include odd cycles and the complete graph on an odd number of nodes.

Remark 13 *Every hypomatchable graph is connected.*

Theorem 14 *A hypomatchable inequality (11) defines a facet of P_S if the following conditions hold:*

(i) $L = E$, $K = K' \cup K''$;

(ii) G is a complete graph and $|V'| < |V| - 1$;

(iii) $G_{F'} = (V(F'), F')$ is hypomatchable.

Proof. Since the conditions of Proposition 4 are satisfied, P_S is fulldimensional.

Consider a facet defining inequality

$$\sum_{e \in E} a_e y_e + \sum_{k \in K} \sum_{e \in E} a_e^k x_e^k \geq \alpha \tag{15}$$

such that each root of (11) satisfies (15) with equality. As P_S is fulldimensional, it suffices to show that the two inequalities (11) and (15) are equal up to a positive scalar multiple.

We first describe a basic construction of roots of (11). Let M be a matching of $G_{F'}$ of size $\lfloor n/2 \rfloor$ or a supermatching of $G_{F'}$ of size $\lceil n/2 \rceil$. Set $y_e = 1$ for $e \in M$ and $y_e = 0$ for $e \in F' \backslash M$. If v_i is incident to at least one edge in M, route k_i such that $x^{k_i}(\delta_G(v_i) \setminus M) = 0$. If v_i is not incident to an edge in M (there can be at most one such node) route k_i such that $x^{k_i}(\delta_G(v_i) \cap F') = 0$. With $L = E$ and G complete it is always possible to find such a routing. Then (x, y) is feasible and a root of (11).

For each $e \in E \setminus F'$ one can construct a root (x, y) of (11) with $y_e = 0$ by choosing a supermatching in the basic root construction and avoiding e in the routing. This works because of $|V'| < |V| - 1$. For each $e \in F' \setminus F$ one can construct a root (x, y) of (11) with $y_e = 0$ by choosing a maximum matching M of $G_{F'}$ that avoids e. That is possible by condition (iii). By comparing these solutions with the corresponding root solutions where y_e is set to 1, one proves that $a_e = 0$ for each $e \in E \setminus F$. Similarly, we derive $a_e^k = 0$ for $e \in E \setminus F$ and for those $k \in K$ whose coefficient in (11) is 0.

For given $e = [v_i, v_j] \in F$ choose a perfect matching of $G_{F'} \setminus \{v_i\}$ and augment it to a supermatching M by adding edge e. With the basic construction one may now create a root with $y_e = 1$, $x_e^{k_i} = 1$ and no other demand using e. By setting $x_e^k = 1$ for some $k \neq k_i$ one obtains a new root. (Note that K contains only small demands, because of the condition $K = K' \cup K''$.) This proves $a_e^k = 0$. Since e and k was arbitrary, and, moreover, $k_i \neq k_j$, one gets $a_e^k = 0$ for all $e \in F$ and $k \in K$.

For $e = [v_i, v_j] \in F' \setminus F$ one can similarly prove that $a_e^k = 0$ for all $k \neq k_i$ $(= k_j)$.

Thus, whenever in (11) a coefficient of some variable is zero, then the corresponding coefficient in (15) is zero.

Let $e = [v_i, v_j] \in F$ and $f \in \delta_G(v_i) \setminus F$. We shall prove that $a_e = a_f^{k_i}$. Pick a perfect matching M in $G_{F'} \setminus \{v_i\}$. If $f \in F'$ augment this matching by edge f. The basic routing construction can be done such that demand k_i is routed on edge f. Compare this solution to the one where demand k_i uses edge e instead of f. We get $a_e = a_f^{k_i}$ for any e, f and k_i chosen as above. If $e \in F' \setminus F$ and $f \in \delta_G(v_i) \setminus F'$, a similar construction shows $a_e^{k_i} = a_f^{k_i}$.

Now let e, f, and g be three edges with endnode v_i. When e and f are in F and $g \notin F$ we have shown $a_e = a_g^{k_i} = a_f$. When $e \in F$, $f \in F' \setminus F$, and $g \notin F'$ we have $a_e = a_g^{k_i} = a_f^{k_i}$. Note that e, $f \in F' \setminus F$ is not possible. By Remark 13 $G_{F'}$ is connected, and thus (15) is a scalar multiple of (11), showing that (11) defines a facet of P_S. □

We note that conditions (i) and (ii) are present only to simplify the proof. Any one of them can be relaxed. Especially the restriction on the number and size of demands is not necessary. Condition (iii) is probably necessary, but we have not been able to prove this.

In our computations we have chosen F to be an odd cycle. We call this subclass of (11) *cycle inequalities*.

The hypomatchable inequalities may be extended into larger classes of facet defining inequalities using lifting techniques. The idea is to shrink certain node sets in some PIPE instance and thereby obtain a "smaller" related instance for which a hypomatchable inequality is valid. The lifted inequality is obtained by letting all edges that were shrunk get a coefficient zero. One can show (under certain conditions on the subgraphs that are shrunk) that a lifted hypomatchable inequality is nonredundant.

4 The cutting plane algorithm

In this section we describe the implementation of our cutting plane algorithm for solving the PIPE problem. We assume that the reader is familiar with the general outline of a cutting plane algorithm (see, for instance, [1] or [17]). The following table presents the main steps of such a cutting plane algorithm. We hereafter let V' be the subset of V consisting of all endnodes of demand

edges (i.e., nodes u^k and v^k for $[u^k, v^k] \in K$).

1. Initialization
2. **while** branch-and-bound tree is not empty
3. select a leaf from the tree
4. **do**
5. solve the LP
6. separate inequalities and add them to the LP
7. **while** there are violated inequalities
8. call primal heuristic
9. branch if necessary
10. print best feasible solution and best lower bound
11. **STOP.**

In the *Initialization* phase we set up the first LP and initialize the branch-and-bound tree with the root node representing the whole problem. As initial cuts for the first LP we use the trivial inequalities $0 \leq x_e^k \leq 1$, $0 \leq y_e \leq 1$, and the degree constraints $x^k(\delta_G(u^k)) \geq 1$ and $x^k(\delta_G(v^k)) \geq 1$ for $[u^k, v^k] \in K$. In addition, we add some of the strengthened cut inequalities in the following way. For each node $v \in V'$, we check whether $(d(\delta_D(v))$ modulo $B) \neq 0$. If that is the case, we add the corresponding strengthened cut inequality to the initial LP. If not, we try to extend the node set $W = \{v\}$ in a greedy like fashion (by checking all neighbouring nodes of W) until we find a set W satisfying $(d(\delta_D(W))$ modulo $B) \neq 0$ (in this case we add the strengthened cut inequality induced by W to the initial LP) or the list of neighbours is empty.

This set of inequalities represents the first LP. For solving the linear programs we use CPLEX[1], a very fast and robust linear programming solver. Step 6, separating inequalities, will be discussed in Subsection 4.1 in detail. We add inequalities $a^T(x, y) \geq \alpha$ to the current LP, if the slack $(= \alpha - a^T(\bar{x}, \bar{y})$, where $(\bar{x}, \bar{y})$ is the current LP solution) is at least VIEPS, which is set to 0.1 in our implementation. In order to keep the LPs of moderate size, each inequality is assigned an "age" (at the beginning the age is set to 0). Each time the inequality is not tight at the current LP solution, the age is increased by one. If the inequality gets too old, i.e., the age exceeds a

[1]CPLEX is a registered trademark of CPLEX Optimization, Inc.

certain limit (in our implementation this limit is set to 8), the inequality is eliminated from the LP.

If we do not find more violated inequalities and there is still a gap between the current optimum LP objective function value and the best known feasible solution, we call the primal heuristic (step 8). This procedure will be described in Subsection 4.2. If (after a possible improvement of the best feasible solution) there is still a gap between the current local lower bound and the best solution, we branch on a variable that is closest to 0.5. In this way, we create two new subproblems, one where the branching variable is fixed to 0, and one, where it is fixed to 1. We add these two subproblems to our branch-and-bound tree and continue with step 3. The strategy we use to select the next leaf is best-first-search, i.e., we select a leaf with the worst lower bound (equal to the global lower bound).

In principle, if we let this algorithm run forever, it will find an optimum solution. But, "forever" really can mean forever. In the next section we will see two examples where we cannot improve the gap between the best lower and upper bound after hours of CPU time. Therefore, the algorithm has an option to stop when a certain time limit or a certain number of branch-and-bound nodes is exceeded. In this case, we print out the best feasible solution and the global lower bound providing a solution guarantee for the best solution.

4.1 Separation algorithms

In the following we discuss separation algorithms for the cut inequalities (1) (i) and (8), the cover inequalities (7), and the cycle inequalities (11). The separation problem for a class of inequalities can be stated as follows:

Separation problem. Given a vector $(\bar{x}, \bar{y})$ with $\bar{x} \in \mathbf{R}^{K \times E}, \bar{y} \in R^E$ and a class of inequalities valid for $P_{\mathcal{S}}$. Decide whether $(\bar{x}, \bar{y})$ satisfies all inequalities of that class, and if not, find an inequality $a^T(x, y) \geq \alpha$ with $a^T(\bar{x}, \bar{y}) < \alpha$.

Since we added to the initial LP all trivial inequalities, we can suppose in the following that all components of the actual LP solution $(\bar{x}, \bar{y})$ are nonnegative and less than or equal to 1.

Cut inequalities. In order to solve the separation problem for the cut inequalities (1) (i) for a particular demand $k \in K$, we have to decide whether

the minimum cut capacity between u^k and v^k is less than 1 where edge capacities are given by $\bar{x}^k$. This can be done using any max-flow algorithm. We implemented the highest-label preflow push algorithm suggested by Goldberg and Tarjan [5] (see also [2]). This algorithm runs in time $O(|V'|^2\sqrt{|E|})$.

We also use our max-flow algorithm to find violated strengthened cut inequalities (8). We assign each edge $e \in E$ the capacity $\max\{0, \bar{y}_e - \sum_{k \in K_2} \bar{x}_e^k\}$, and determine for each $[u^k, v^k] \in K_1$ a minimum $[u^k, v^k]$-cut, $\delta_G(W^*)$ say, for $W^* \subset V$. If $|\delta_D(W^*) \cap K_1|/B$ is not integer, we add the corresponding strengthened cut inequality in case it is violated.

Cover inequalities. Separating the cover inequalities (7) is easy. Since there are linearly many (linear in the number of demands and edges, see Section 3), we sequentially check all of them for possible violation. One might think it is possible to store all cover inequalities explicitly in the LP, but it turns out that the number of such inequalities may be very large (see next section), though only a fraction of them is needed to solve the problem.

Cycle inequalities. We do not know whether the separation problem for the cycle inequalities (11) can be solved in polynomial time (when F defines a cycle), because we do not know how to find a low-weight cycle that only contains terminal nodes for demands of value 1. In the following we present a heuristic, where we first try to find a minimum cycle with respect to a certain weight function, and then choose, if possible, for each node in the cycle, the best demand of size 1.

Let us first describe the weight function according to which we want to find a minimum cycle. Consider again (11) with F being a cycle with node set $\{v_1, \ldots, v_n\}$, n odd, and $k_1, \ldots, k_n$ demands of size 1 such that demand k_i has an endnode at node v_i $(i = 1, \ldots, n)$. Since the objective function is nonnegative, the optimum integer solution and also the current LP solution certainly satisfy $x^{k_i}(\delta(v_i)) = 1$. If we subtract this equation (for $i = 1, \ldots, n$) from the cycle inequality, we get

$$y(F) - \sum_{k \in K_2} x^k(F) - \sum_{i=1}^{n} x^{k_i}(\delta_G(v_i) \cap F') + \sum_{k \in K''} \sum_{\substack{e \in E: \\ e \text{ connects } u^k, v^k}} \cdot x_e^k \geq -\lfloor \frac{n}{2} \rfloor. \quad (16)$$

Suppose for a moment that we do not have parallel edges, and all demands

$k_1, \ldots, k_n$ are different. If we choose as edge weights, for $uv \in E$,

$$w_{uv} := 0.5 + \bar{y}_{uv} - \sum_{k \in K_2} \bar{x}^k_{uv} - \sum_{\substack{k \in K_1: \\ \{u^k, v^k\} \cap \{u,v\} \neq \emptyset}} \bar{x}^k_{uv},$$

then $w(F) - 0.5$ exactly coincides with the difference between the left- and right-hand sides of (16), when F is an odd cycle and each node in the cycle has exactly one incident demand of size 1. Thus, if $w(F) < 0.5$ and F is odd, we have a violated cycle inequality, otherwise not. If there are parallel edges $\{e_1, \ldots, e_p\}$, $p \geq 2$, connecting nodes u and v, we aggregate these edges to a single edge, uv say, and assign it the weight

$$w_{uv} := 0.5 + \sum_{i \in I} \Big(\bar{y}_{e_i} - \sum_{k \in K_2} \bar{x}^k_{e_i} - \sum_{\substack{k \in K_1: \\ \{u^k, v^k\} \cap \{u,v\} \neq \emptyset}} \bar{x}^k_{e_i} \Big),$$

where $I := \{i \in \{1, \ldots, p\} \mid \text{there exists some } k \in K_1 \text{ with } \bar{x}^k_{e_i} > 0\}$.

Now we determine a minimum cycle F^* with respect to edge weights w (see below). Note that according to the definition of w, $w(F^*) - 0.5$ is a lower bound for the slack of the most violated cycle inequality. Thus, if $w(F^*) \geq 0.5$, there is no violated cycle inequality. Otherwise, if all nodes in the cycle are incident to some demand of size 1 (otherwise the heuristic fails), we run through all nodes of the cycle and determine for each node v_i the best possible demand k_i, i.e., we choose

$$k_i := \operatorname*{arg\,min}_{k \in K_1 : v_i \in \{u^k, v^k\}} \bar{x}^k(\delta_G(v_i) \setminus F^*).$$

It might be that the cycle inequality that is defined by our choices F^* and $k_1, \ldots, k_{|F^*|}$ yields no longer a violated inequality, because the edge weights w_{uv} do not reflect the exact slack, when nodes have more than one incident demand of size 1.

There remains the problem of finding a minimum cycle F^* in an undirected graph $G = (V, E)$ with edge weights w_{uv}, $uv \in E$, that can be negative. If the edge weights are indeed arbitrary, the problem to determine a minimum cycle is $\mathcal{NP}$-complete. However, we can decide whether there is a cycle of negative weight and if not, find a minimum cycle by transforming the problem to a perfect matching problem (see [2]). Thus, we can decide whether there exists a cycle of weight less than 0.5 which might give rise to a

violated inequality. Since a perfect matching algorithm is very time consuming and since such an algorithm might return just one cycle, we preferred to implement the following heuristic. Starting from each node $v \in V$, we determine a shortest spanning tree by using Prim's algorithm ([16]) and check all fundamental cycles whether their weight is less than 0.5. In case the cycle is even, we contract one edge. This results in many violated cycle inequalities, and for many instances this algorithm finds a cycle of weight less than 0.5 whenever there is one.

4.2 The primal heuristic

In step 8 of our branch-and-cut algorithm we call the primal heuristic. We do that after the cutting plane phase for the current node is finished, i.e., we have not found any more violated inequalities, and the current LP solution is fractional. The idea of our primal heuristic is to fix a set of fractional variables to zero or one, solve the LP again, and iterate this process until all variables are integer. This heuristic idea, often used in general mixed integer programming solvers, is sometimes called "plunging" or "diving", because we "dive" deep into the branching tree and "plunge" for a feasible solution. The tuning parameters of this heuristic are the order in which the fractional variables should be fixed in one step and their number. We performed several tests trying to give an answer to these two questions. It turned out that the heuristic in general worked best when we just fix one fractional variable at a time and choose a fractional variable that is close to one. Moreover, we fix all variables that are 1 to value 1 for the rest of the heuristic.

. If the heuristic does not change the linear program (except for fixing variables) it frequently ends with an integral solution that violates one of the cut or cover constraints in (1), since not all of these constraints are contained in the LP. Therefore we separate those inequalities for each fractional solution appearing in the course of the heuristic. This unfortunately slows the heuristic down. In order to speed up the separation process in the heuristic, we only add those cut inequalities (1) (i), knapsack inequalities (1) (ii) and cover inequalities (7) that are violated by at least 0.5 (the usual violation epsilon in the cutting plane phase is 0.1). Moreover, we restrict the number of times the heuristic is called, depending on its success. More precisely, we calculate (i) the ratio between the time spent in the heuristic and the total time, and (ii) the ratio between the number of times the heuristic could improve the

best solution and the number of times the heuristic was called. If the "time" ratio is less than the "success" ratio, we call the heuristic, otherwise not. The results in the next section show that this strategy performs quite well, we obtain reasonably good primal solutions by spending at most 30% (usually less than 10%) of the total time in the heuristic.

5 Computational results

In this section we report on the test runs performed with our branch-and-cut algorithm. The code is implemented in C, and all results were obtained on a Sun SPARC 20 Model 71. The examples are modified real-world examples with pipe capacity $B = 4$. Table 1 summarizes the data. Column 2 and 3 show the number of nodes and links of the physical network, Columns 4 and 5 contain the corresponding information for the pipe graph. Here, $|V'|$ is the number of nodes incident to some demand edge. Columns 6 and 7 give the number of demands of size 1 and 4.

The last column gives the number of 0/1 variables in our IP formulation. The numbers range from about 250 for the smallest problem up to 25000 variables. The test series in Table 1 are based on two physical networks. *nw* is an example approximating parts of the physical network in Norway. All other examples whose name starts with "nw" are derived from *nw*. *nw3* differs from *nw* in that it contains some further physical links, that some more physical nodes are endnodes of demands, and that the set of possible express pipes is extended. The remaining "nw"-examples are variations of these two instances, where we wanted to test how sensible the solution is with respect to changes in the input data. If the name contains the letters ".0", the link capacities (of the example without ".0") are multiplied by ten in order to see what influence the link capacities have on the solution. Examples ending with ".p" have more express pipes than the corresponding example without ".p". The input pipes in the ".p"-examples were generated by finding for each demand k a set of short $u^k v^k$-paths (these were determined by adding certain edges to shortest path trees). Example *nw3.d1.p* results from *nw3.p* by changing the size of 9 demands from B to 1. The last three examples in Table 1 are typical for local area networks. The demand graph of *terbstar* consists of node-disjoint stars. The demands of *terbco* form a complete graph between the root nodes of these stars, and the demand graph of *terbstco* is

the union of these two demand graphs.

Example	N		G		Demands		Variables
	$\|V\|$	$\|L\|$	$\|V'\|$	$\|E\|$	size 1	size B	
nw	27	44	5	22	2	8	242
nw.p	27	44	5	63	2	8	693
nw3	27	60	10	91	3	18	2002
nw3.p	27	60	10	191	3	18	4202
nw3.0	27	60	10	91	3	18	2002
nw3.0.p	27	60	10	191	3	18	4202
nw3.d1.p	27	60	10	191	12	9	4202
terbco	62	81	7	113	0	21	2486
terbstar	62	81	56	248	36	12	12152
terbstco	62	81	56	359	36	33	25130

Table 1: Input data

Unfortunately, the network planners could not give us any reasonable numbers for the cost of installing the express pipes. Thus, we played with this parameter a little bit and performed different tests varying the installation cost γ from 0 (which means that we get the express pipes for free) up to 10 which results in rather high express pipe costs compared to the routing costs.

An interpretation of γ may be illustrated as follows. If we have the choice between installing a new direct pipe for a demand of value 1 and using the spare capacity of an existing path of "length" less than γ in the physical network, then the "long" path is preferred.

Table 2 through 5 summarize our tests. Column 2 gives the number of inequalities of the initial LP, Columns 3 to 5 show the number of violated cut (those of type (1) (i) and (8) together), cover and cycle inequalities. The number of LPs solved (including those in the primal heuristic) and the number of solved branch-and-bound nodes are presented in Columns 6 and 7. Columns 8 and 9 show the global lower bound and the value of the best feasible solution after the algorithm stopped. The total time (in CPU seconds) of the algorithm and the time spent in the heuristic are given in the last two columns.

Looking at Table 2 with the results for $\gamma = 0$ we see that we can solve

all problem instances in the root node, i.e., we do not have to branch. Even more, with the exception of *terbstar* and *terbstco* the solutions of the root LPs are integer, since the primal heuristic has not been called. This indicates that the inequalities we separate are indeed important to solve the problems. Note that all ".p"-examples have lower objective function value than their corresponding counter part without ".p". An interesting question is how the number and variability of the express pipes influences the solutions. To completely answer this question and to find the best feasible solution among all possible express pipes, our algorithm must be embedded into a column generation approach. In case the network planners do not impose any restrictions on the set of express pipes, it will be a challenge for the future to integrate the cutting plane and the column generation approach in order to obtain the globally best solution. Whether the link capacities have an influence on the quality of the solution, we cannot draw any conclusions from this test set. For *nw3* the optimum is the same, for *nw3.p* we obtain a better solution. A noteworthy fact is that all "nw"-examples are solved within seconds. The "terb"-examples seem to be harder, but still our algorithm provides the optimum solution after at most 25 minutes of CPU time.

For $\gamma = 1$ (see Table 3) the results are basically the same with mostly slightly higher running times. But, if we further increase γ the situation changes (Table 4 and 5). We still can solve all "nw"-examples within one minute, but for *terbstar* and *terbstco* our algorithm gets stuck. We can give a solution guarantee of 9% or less after about 3 hours of CPU time (which might be acceptable in practice), but we almost cannot improve this gap any further, even if we spent some more hours of CPU time. Since the express pipes are very expensive, the algorithm tries to avoid using y-variables. What is missing are further inequalities (like the hypomatchable inequalities) that force the y-variables to one whenever the routing variables x are positive. If network planners will indeed come up with such high express pipe installation costs and they are really interested in finding the optimum solution more research in this area will be necessary.

Example	Cutting Planes				lps	bab	lb	ub	Times (sec)	
	init	knap	cuts	cycle					Heur	Total
nw	25	28	6	0	5	1	183	183	0.0	0.1
nw.p	25	71	0	0	11	1	156	156	0.0	0.2
nw3	52	125	30	18	8	1	287	287	0.0	0.6
nw3.p	51	224	3	17	19	1	243	243	0.0	1.7
nw3.0	52	63	33	4	12	1	287	287	0.0	0.5
nw3.0.p	51	46	1	4	4	1	227	227	0.0	0.4
nw3.d1.p	51	469	7	108	13	1	139	139	0.0	5.0
terbco	42	31	31	0	8	1	536	536	0.0	0.5
terbstar	150	1200	389	334	42	1	193	193	1.0	74.8
terbstco	192	3683	1194	1142	138	1	732	732	17.8	1483.0

Table 2: $\gamma = 0$

Example	Cutting Planes				lps	bab	lb	ub	Times (sec)	
	init	knap	cuts	cycle					Heur	Total
nw	25	24	7	0	6	1	197	197	0.0	0.1
nw.p	25	57	3	0	9	1	166	166	0.0	0.2
nw3	52	120	34	34	8	1	315	315	0.0	0.7
nw3.p	51	226	8	27	14	1	264	264	0.0	1.8
nw3.0	52	113	33	33	7	1	315	315	0.0	0.8
nw3.0.p	51	146	5	21	8	1	248	248	0.0	0.9
nw3.d1.p	51	532	14	112	29	1	156	156	0.1	8.6
terbco	42	47	32	0	10	1	557	557	0.0	0.6
terbstar	150	1385	384	511	23	1	241	241	0.6	107.0
terbstco	192	3431	1087	1056	93	1	801	801	2.5	1298.7

Table 3: $\gamma = 1$

Example	Cutting Planes				lps	bab	lb	ub	Times (sec)	
	init	knap	cuts	cycle					Heur	Total
nw	25	24	7	0	6	1	253	253	0.0	0.1
nw.p	25	60	3	0	10	1	206	206	0.0	0.2
nw3	52	130	31	28	9	1	427	427	0.0	0.9
nw3.p	51	295	9	50	19	1	348	348	0.0	4.2
nw3.0	52	137	35	38	9	1	427	427	0.0	1.9
nw3.0.p	51	235	6	46	20	1	332	332	0.0	2.5
nw3.d1.p	51	757	15	159	31	1	217	217	0.0	16.9
terbco	42	56	26	0	8	1	641	641	0.0	0.5
terbstar	150	7821	3417	3202	3064	697	423	433	965.0	10003.3
		12821	5194	5138	4950	1303	423	433	1386.9	20008.6
		26186	10967	11332	9679	2632	424	433	1817.9	40014.0
terbstco	192	5365	1596	1833	1087	196	1064	1077	1320.2	10141.5
		7270	2295	2643	1976	482	1065	1077	1894.6	20123.8
		11938	4057	4885	3941	1126	1066	1077	2617.5	40000.8

Table 4: $\gamma = 5$

Example	Cutting Planes				lps	bab	lb	ub	Times (sec)	
	init	knap	cuts	cycle					Heur	Total
nw	25	25	9	0	7	1	323	323	0.0	0.1
nw.p	25	76	3	0	16	1	256	256	0.0	0.3
nw3	52	199	42	39	25	1	567	567	0.0	3.9
nw3.p	51	442	17	167	88	1	453	453	0.0	16.4
nw3.0	52	184	42	36	20	1	567	567	0.0	3.7
nw3.0.p	51	389	16	168	100	1	437	437	0.0	15.7
nw3.d1.p	51	1078	35	259	78	1	292	292	2.4	56.3
terbco	42	64	25	0	8	1	746	746	0.0	0.5
terbstar	150	4840	5024	1518	1582	81	626	678	1038.5	10002.3
		8732	10460	2883	2903	193	629	678	1464.9	20028.5
		15728	19508	5491	5084	383	632	675	2888.8	40007.1
terbstco	192	4976	1330	1085	777	17	1370	1424	4143.3	10096.8
		6278	2813	1657	1450	58	1374	1424	5438.5	20166.5
		8773	5863	2575	2375	127	1378	1424	7263.0	40060.3

Table 5: $\gamma = 10$

6 Conclusions

Pipe selection and routing problems in telecommunications is a rather new area for applied combinatorial optimization. There are many related problems and models to the one in our study that call for analysis and algorithmic development.

The results obtained in our study indicate that a cutting plane approach may be attractive for solving certain real-world pipe selection and routing problems. Further work could be directed towards solving problems with larger pipe installation cost γ. This would require a further polyhedral study of the polytope P_S and, for this, other inequalities derived from path packing problems (see [6], [11]) may be of interest.

A very interesting area is to extend the problem by allowing (almost) arbitrary express pipes and to modify the cutting plane algorithm by adding a column generation scheme.

Acknowledgement. This research was financed by *Telenor AS Research and Development* (Project number TFN9506A).

References

[1] D. Applegate, R.E. Bixby, V. Chvátal, W. Cook, Finding cuts in the TSP, DIMACS Technical Report 95-05, March 1995.

[2] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, *Network flows: theory, algorithms, and applications*, Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

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

[4] G. Chartrand and L. Lesniak, *Graphs and digraphs.* Wadsworth and Brooks, California, 1986.

[5] A. V. Goldberg, R. E. Tarjan, A new approach to the maximum flow problem, *Journal of ACM* 35, 921 – 940, 1988.

[6] M. Grötschel, A. Martin, R. Weismantel, *Packing Steiner trees: polyhedral investigations*, Preprint, Konrad-Zuse-Zentrum (ZIB), Berlin, SC 92-8, 1992, to appear in *Mathematical Programming.*

[7] M. Grötschel, A. Martin, R. Weismantel, *Packing Steiner trees: a cutting plane algorithm and computational results*, Preprint, Konrad-Zuse-Zentrum (ZIB), Berlin, SC 92-9, 1992, to appear in *Mathematical Programming.*

[8] M. R. Kramer, J. van Leeuwen, The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits, in F. P. Preparata (ed.), *Advances in Computing Research*, Vol. 2: VLSI theory, Jai Press, London, 129 – 146, 1984.

[9] R. Lorentzen, Mathematical methods and algorithms in the network utilization planning tool RUGINETT. *Telektronikk* 90 (4):73–82, 1994. (Telenor Research, P.O.Box 83, 2007 Kjeller, Norway)

[10] L. Lovász and M. D. Plummer, *Matching theory*, North-Holland, 1986.

[11] A. Martin, *Packen von Steinerbäumen: Polyedrische Studien und Anwendungen*, Ph.D.Thesis, Konrad-Zuse-Zentrum (ZIB), Berlin, TR 92-4, 1992.

[12] K. Menger, *Zur allgemeinen Kurventheorie*, Fundamenta Mathematicae 10, 96-115, 1927.

[13] G. Nemhauser and L.A. Wolsey, *Integer and combinatorial optimization* Wiley, 1988.

[14] Kyungchul Park and Seokhoon Kang and Sungsoo Park, An integer programming approach to the bandwidth packing problem, unpublished, Dept. of Industrial Engineering, Korea Advanced Institute of Science and Technology, Taejon, Korea, 1994.

[15] M. Parker and J. Ryan, A Column Generation Algorithm for Bandwidth Packing, *Telecommunications Systems* 2, 185–195, 1994.

[16] R. C. Prim, Shortest connection networks and some generalizations, *Bell System Technical Journal* 36, 1389 – 1401, 1957.

[17] M. Padberg, G. Rinaldi, A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems, *SIAM Review* 33, 60 – 100, 1991.

[18] A. Schrijver, *Theory of linear and integer programming*, Wiley, Chichester, 1986.

[19] R. Weismantel, On the 0/1 knapsack polytope, Konrad-Zuse-Zentrum (ZIB), Berlin, Preprint SC 94-1, 1994.