COMPUTATIONAL RESULTS WITH A CUTTING PLANE ALGORITHM FOR DESIGNING COMMUNICATION NETWORKS WITH LOW-CONNECTIVITY CONSTRAINTS

MARTIN GRÖTSCHEL
Konrad-Zuse-Zentrum für Informationstechnik, Berlin, Germany

CLYDE L. MONMA
Bell Communications Research, Morristown, New Jersey

MECHTHILD STOER
Konrad-Zuse-Zentrum für Informationstechnik, Berlin, Germany

(Received October 1989; revision received January 1991; accepted June 1991)

We describe a cutting plane approach to the problem of designing survivable fiber optic communication networks. This problem can be formulated as a minimum cost network design problem with certain low-connectivity constraints. Computational results on real-world telephone network design problems demonstrate the effectiveness of our cutting plane method. The facet-inducing inequalities for the convex hull of the solutions to this problem on which our algorithm is based are studied in detail in a companion paper.

Survivability is a particularly important issue for fiber optic communication networks. The high capacity of fiber facilities results in much more sparse network designs with larger amounts of traffic carried by each link than is the case with traditional bandwidth-limited technologies. This increases the potential damage to network services due to link or node failures. It is necessary to tradeoff the potential for lost revenues and customer goodwill against the extra costs required to increase the network survivability. Recent work on methods for designing survivable fiber communication networks by Cardwell, Monma and Wu (1989), and Monma and Shallcross (1989) concludes that "two-connected" topologies provide a high level of survivability in a cost effective manner, and that good heuristic methods exist for quickly generating "near-optimal" networks. In particular, it was determined that a network topology should provide for at least two diverse paths between certain "special" offices, thus providing for protection against any single link or single node failure for traffic between these offices. These special offices represent high revenue producing offices and other offices that require a higher level of network survivability.

Our paper presents a so-called cutting plane algorithm for computing a minimum cost network that satisfies the survivability conditions described above. The costs considered here are only those needed for establishing the network topology, such as placing conduits in which to lay fiber cables, placing the cables into service, and other related costs. We do not consider routing, multiplexing, and repeater costs. Computational results show that our algorithm can compute, within a few minutes, minimum cost survivable telephone networks of the type and size arising at Bell Communications Research. We could also show that the heuristic solutions reported by Monma and Shallcross (on certain real-world instances of telephone network design) are actually near-optimal.

The cutting plane approach used in our algorithm is based on optimization over the convex hull of the solutions to the survivable network design problem, the so-called **2ECON** or **2NCON** polyhedron. A general, integer linear programming approach to network design problems with connectivity requirements is presented in Grötschel, and Monma (1990) along with a preliminary study of these problems from a polyhedral point of view. Several classes of facet-defining inequalities for the **2ECON** and **2NCON** polyhedron were identified in our companion paper (Grötschel, Monma and Stoer 1989). The present paper is based on the theory developed in Grötschel, Monma and

Subject classifications: Communications, designing optic-based survivable communication systems. Programming, integer: cutting plane/facet generation.
Area of review: TELECOMMUNICATIONS.

Stoer (1989), and we shall make several references to these results in what follows. A special case, where at least two edge-disjoint paths are required between *all* pairs of offices, is investigated by Cornuéjols, Fonlupt and Naddef (1988), Mahjoub (1988), and Monma, Munson and Pulleyblank (1990) from the polyhedral point of view.

Section 1 introduces graph-theoretical notation and an integer linear programming model of the survivable network design problem with low-connectivity constraints. In Section 2, we describe an approach for decomposing the network design problem into smaller problems that can be solved independently of each other. In Section 3, we summarize the classes of facet-inducing inequalities from Grötschel and Monma, and Grötschel, Monma and Stoer (1989) that we use in our implementation. The details of the implementation and the heuristic and exact separation algorithms used to generate violated inequalities are described in Section 4. Computational results on real-world telephone network design problems are presented in Section 5, and compared to results obtained by the heuristic methods in Monma and Shallcross.

1. NOTATION AND DEFINITION OF THE ASSOCIATED POLYHEDRA

The problem of designing survivable fiber optic communication networks can be modeled as a minimum cost network design problem with certain low-connectivity constraints. More precisely, we are given a *graph* $G = (V, E)$, where V is a set of *nodes* that represents offices that must be interconnected by a network, and E is a collection of *edges* that represent the possible pairs of nodes between which a direct transmission link can be placed. The graph G may have parallel edges but contains no loops. Each edge $e \in E$ has a nonnegative *fixed cost* c_e of establishing the direct link connection. The cost of establishing a network $N = (V, F)$ consisting of a subset $F \subseteq E$ of edges is $c(F) := \sum_{e \in F} c_e$, the sum of the costs of the individual links contained in F. The goal is to build a minimum cost network so that certain survivability conditions, which we describe below, are satisfied.

The *survivability conditions* require that the network satisfy certain edge and node connectivity requirements. In particular, a nonnegative integer r_s is associated with each node $s \in V$ that represents its *connectivity requirement*. This means that, for each pair of distinct nodes $s, t \in V$, the network $N = (V, F)$ to be designed has to contain at least

$$r(s, t) := \min\{r_s, r_t\}$$

edge-disjoint (or node-disjoint) $[s, t]$-paths. We call r_s the *connectivity type* of node s, or the *type* of node s.

In the remainder of this paper we consider the important, practical case where the connectivity requirements satisfy $r_s \in \{0, 1, 2\}$ for all $s \in V$. This includes the problem of designing survivable fiber optic telephone networks (Cardwell et al. 1988, Cardwell, Monma and Wu 1989, and Monma and Shallcross 1989). We define the **2ECON** (respectively, **2NCON**) problem to be the network design problem where edge-disjoint (respectively, node-disjoint) paths are required, and we will speak in this case of two-connected edge (or node) survivability constraints. We will say that a **2ECON** or a **2NCON** problem is given by (G, r) and implicitly assume that $G = (V, E)$ is a graph and r a vector of node types with $r \in \{0, 1, 2\}^V$.

Given a graph $G = (V, E)$ and $W \subseteq V$, the edge set

$$\delta(W) := \{ij \in E \mid i \in W, j \in V \setminus W\}$$

is called the *cut* (induced by W). (We will write $\delta_G(W)$ to make clear—in case of possible ambiguities—with respect to which graph the cut induced by W is considered.) For $W, W' \subseteq V$ with $W \cap W' = \phi$, we define $[W : W'] := \{ij \in E \mid i \in W, j \in W'\}$, so $\delta(W) = [W : V \setminus W]$. We write $\delta(v)$ for $\delta(\{v\})$ if v is a single node; W and $V \setminus W$ are called the *shores* of the cut $\delta(W)$.

For $W \subseteq V$, we set $E(W) := \{ij \in E \mid i, j \in W\}$, and denote by $G[W] := (W, E(W))$ the subgraph induced by W. We denote by G/W the graph in which $W \subseteq V$ is shrunk to a node. $G - v$ denotes the graph obtained by removing the node v and all incident edges from G, and $G - F$ denotes the graph obtained by removing the edge set F from G (we write $G - f$ instead of $G - \{f\}$). If e is an edge, so that $G - e$ has more connected components than G, we will call e a *bridge* of G. Similarly, if S is a node set, so that $G - S$ has more connected components than G, we call S an *articulation set* of G. If a single node forms an articulation set, the node is called an *articulation node.*

We extend the connectivity requirement function r to functions operating on sets by setting for all $W \subseteq V, \phi \neq W \neq V$,

$$r(W) := \max\{r_s \mid s \in W\} \quad \text{for all } W \subseteq V, \text{ and}$$

$$\text{con}(W) := \max\{r(s, t) \mid s \in W, t \in V \setminus W\}$$

$$:= \min\{(r(W), r(V \setminus W)\}.$$

Let us now introduce, for each edge $e \in E$, a variable x_e and consider the vector space $\mathbb{R}^E$. Every subset $F \subseteq E$ induces an *incidence vector* $X^F = (X_e^F)_{e \in E} \in \mathbb{R}^E$ by setting $X_e^F := 1$ if $e \in F$, and $X_e^F := 0$ otherwise;

vice versa, each 0/1-vector $x \in \mathbb{R}^E$ induces a subset $F^x := \{e \in E \mid x_e = 1\}$ of the edge set E of G. For any subset of edges $F \subseteq E$, we define $x(F) := \sum_{e \in F} x_e$. We can now formulate the **2NCON** network design problem introduced above as the following integer linear program.

$$\text{Minimize} \sum_{ij \in E} c_{ij} x_{ij}$$

subject to

$$\begin{array}{lll}
x(\delta(W)) \geqslant \mathrm{con}(W) & \text{for all } W \subseteq V, & \\
 & \phi \neq W \neq V; & \text{(1a)} \\
x(\delta_{G-z}(W)) \geqslant 1 & \text{for all } z \in V, \text{ and} & \\
 & \text{for all } W \subseteq V \backslash \{z\} & \\
 & \phi \neq W \neq V \backslash \{z\} & \\
 & \text{with } r(W) = 2 \text{ and} & \\
 & r(V \backslash (W \cup \{z\})) = 2; & \text{(1b)} \\
0 \leqslant x_{ij} \leqslant 1 & \text{for all } ij \in E; & \text{(1c)} \\
x_{ij} \text{ integral} & \text{for all } ij \in E. & \text{(1d)}
\end{array}$$

It follows from Menger's Theorem that, for every feasible solution x of (1), the subgraph $N = (V, F^x)$ of G defines a network satisfying the two-connected node survivability requirements. Removing (1b) results (again by Menger's Theorem) in solutions that satisfy the two-connected edge survivability requirements; i.e., we have an integer linear program for the **2ECON** network design problem. An inequality of type (1a) will be called a *cut inequality*, one of type (1b) is called a *node cut inequality*, and one of type (1c) is called a *trivial inequality*. In Section 4, we note that the cut and node cut inequalities can be checked in polynomial time, and we show that the separation problem is NP-hard for all other classes of inequalities considered here.

The main objective of this paper is to describe a cutting plane approach for the **2ECON** and **2NCON** network design problems, respectively. To do this, we define the following polytopes. Let $G = (V, E)$ and $r \in \{0, 1, 2\}^V$, the vector of node types, be given, then

$$\mathbf{2NCON}(G; r) := \mathrm{conv}\{x \in \mathbb{R}^E \mid x \text{ satisfies (a), (b), (c), (d) of (1)}\}$$

$$\mathbf{2ECON}(G; r) := \mathrm{conv}\{x \in \mathbb{R}^E \mid x \text{ satisfies (a), (c), (d) of (1)}\}$$

are the polytopes associated with the **2NCON** and **2ECON** network design problems. (In these definitions, *conv* denotes the convex hull operator.) We will call these the **2NCON** and **2ECON** *polytopes.*

Cornuéjols, Fonlupt and Naddef study the dominant of the **2ECON**(G; r) polytope in the special case where $r = 2 \cdot \mathbf{1}$ (**1** is the vector with all components equal to 1). Monma, Munson and Pulleyblank study the **2ECON**(G; r) and **2NCON**(G; r) polytope in the special case where $r = 2 \cdot \mathbf{1}$, and G is a complete graph with the edge weights satisfying the triangle inequality; they show that, in this case, there is an optimal solution to **2ECON** that is also feasible for **2NCON**, and they give a certain type of "characterization" of the optimal solutions. Mahjoub found that inequalities (1a) and (1c) are sufficient to characterize the **2ECON**(G; r) polytope, where $r = 2 \cdot \mathbf{1}$ and G is a series-parallel graph. He also describes a class of inequalities for the **2ECON**(G; r)-polytope for general graphs (and $r = 2 \cdot \mathbf{1}$).

2. DECOMPOSITION

In this section, we describe an approach for decomposing the network design problem into smaller problems that can be solved independently of each other. This is especially useful for the sparse graphs of the real-world communication network design problems that we have encountered. Sparse graphs may contain edges that always have to be used by feasible solutions of **2ECON** or **2NCON**, bridges, for instance. If such edges exist (and also in other cases) it is possible to decompose the problem into several subproblems that can be solved independently of each other.

Reducing the problem size by decomposition resulted in substantial reductions in the running time of our algorithm; see Section 5.

Another advantage of decomposition is that we may confine our polyhedral studies to those network design problems that are defined on "nondecomposable" graphs. This implies that the investigated polyhedra are fully dimensional. Details are given in Grötschel and Monma.

Next we will list some situations where decomposition can be applied for a **2ECON** or a **2NCON** problem given by (G, r).

1. For **2ECON** and **2NCON**: if graph G has a bridge or an articulation node.
2. For the **2NCON** problem: if G has an articulation set of size 2 separating two nodes of type 2, or for the **2ECON** problem: if G contains a cutset of two edges separating two nodes of type 2.
3. For **2ECON** and **2NCON**: if G contains an articulation set of size 2 separating a node of type 1 from a node of type 2, but not separating two nodes of type 2. (Decomposition in this case results in

subproblems that cannot be solved independently of each other, but rather can be solved in a recursive manner.)

Before going into the details of these decompositions, let us mention a strategic choice of our implementation. In a first preprocessing stage, our algorithm tries to find articulation nodes and bridges to decompose the original problem into subproblems. These are then solved independently of each other. For some of the decompositions listed above we have implemented only special cases so far.

2.1. Bridges and Articulation Nodes

Let the graph $G = (V, E)$ and the node types $r \in \{0, 1, 2\}^V$ be given. Suppose that G has a bridge $e = v_1v_2$, and $G - e$ decomposes into two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $v_1 \in V_1$ and $v_2 \in V_2$. If $r(V_1) = r(V_2) = 2$, the **2ECON** problem has no solution. The case $r(V_1) = 0$ or $r(V_2) = 0$ is also trivial. In any other case, define new node types r^1 for G_1 by setting $r^1(v_1) := \max\{r(v_1), 1\}$ and $r^1(v) := r(v)$ for all other nodes of G_1. For G_2 we define node types r^2 in the same way. Clearly, if $C_1 \subseteq E_1$ is a feasible solution to **2ECON** on (G_1, r_1) and $C_2 \subseteq E_2$ is a feasible solution to **2ECON** on (G_2, r_2), then $C_1 \cup C_2 \cup \{e\}$ is a feasible solution to the original **2ECON** problem on graph G. The reverse also holds. So we can solve **2ECON** on (G_1, r^1) and (G_2, r^2) independently of each other to find a solution for **2ECON** on (G, r).

The same (trivial, but useful) idea can be applied if G has an articulation node v.

2.2. Articulation Sets of Size 2 Separating Two Nodes of Type 2

For the **2NCON** problem it is also useful to look for articulation sets consisting of two nodes $\{u, v\}$, whose removal from G creates at least two components that each contain type two nodes. Let $(\bar{V}_i, \bar{E}_i)$, $i = 1, \ldots, p$ be the p different components of $G - \{u, v\}$. Define $G_i = (V_i, E_i)$, $i = 1, \ldots, p$ as the graph obtained from $(\bar{V}_i, \bar{E}_i)$ by adding nodes u and v, the edge set $[\{u, v\}:V_i] \cap E$, and an artificial edge $e_i := uv$ with cost 0. (Note that an edge between u and v in G will be contained in an optimum solution if and only if its cost is negative.) For G_i we define a vector r^i of node types by setting $r^i_s := r_s$ for all nodes $s \in \bar{V}_i$, and

$$r^i_u := \begin{cases} 2, & \text{if } r(\bar{V}_i) = 2, \\ \max\{r_u, 1\}, & \text{if } r(\bar{V}_i) = 1, \\ 0, & \text{if } r(\bar{V}_i) = 0; \end{cases}$$

r^i_v is defined analogously.

Clearly, if C is feasible for **2NCON** on (G, r), then $C_i := (C \cap E_i)$ plus the artificial edge is feasible for **2NCON** on (G_i, r^i) for $i = 1, \ldots, p$. Conversely, if $C_i \subseteq E_i$, $i = 1, \ldots, p$ is feasible for **2NCON** on (G_i, r^i), then $\bigcup_{i=1}^{p} (C_i \cap E)$ is feasible for **2NCON** on (G, r). So we can solve the p subproblems defined on (G_i, r^i) to derive a feasible solution to **2NCON** on the whole graph G. (See Figure 1. Here and in all other figures, big squares denote node sets W with $r(W) = 2$, and big circles denote node sets W with $r(W) = 1$. Nodes of type 1 or 2 are denoted by small circles and squares, respectively. A node of type 0 is depicted only by its name without any symbol.)

Note that this decomposition (using an articulation set of size 2) is, in general, infeasible for the **2ECON** problem. But, if there exists a cut of two edges $\{e, f\}$ so that in $G - \{e, f\}$ two nodes of type 2 are disconnected, both of these edges have to be used by any feasible solution, and we can decompose the **2ECON** problem on G in the same manner as above.

So far, we have implemented only a special case of this type of decomposition. It is the case where G contains a node w of type 2 and exactly two neighbor nodes u and v. If there is still one more node of type 2 besides u, v, and w, the decomposition described above can be applied to the articulation set $\{u, v\}$, that is, we replace the edges uw and wv by a single edge uv with cost 0.

2.3. Articulation Sets of Size 2 Separating Two Nodes of Type ≥ 1

The following decomposition works for the **2ECON**, **2NCON**, and even the Steiner tree problem. It is motivated by a similar sort of decomposition for the

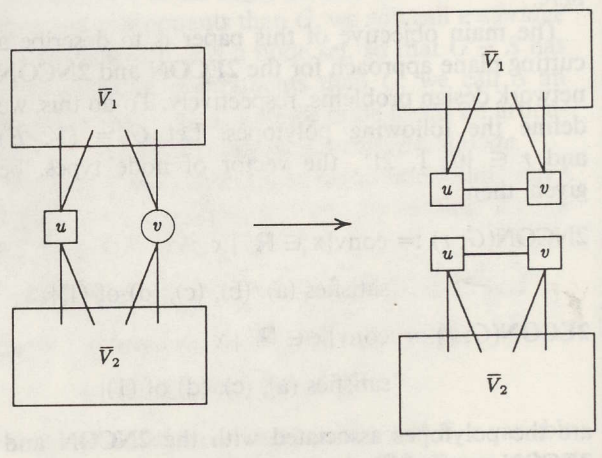

Figure 1. Decomposition using an articulation set of size 2.

Steiner tree problem on directed graphs described by Prodon, Liebling and Gröflin (1985).

The decomposition to be described here can be used in the situation where G contains an articulation set $\{u, v\}$ separating a node of type 1 from a node of type at least 1, but not separating two nodes of type 2. Actually, there are three decompositions according to the node types of u and v, namely for the cases that:

1. $r_u = r_v = 0$,
2. $r_u \geq 1$ and $r_v = 0$,
3. $r_u \geq 1$ and $r_v \geq 1$.

A characteristic of these decompositions is that the subproblems must be solved in a certain order, not independently of each other, because the output of several of these subproblems determines the input to the last one.

2.3.1. Both Nodes in the Articulation Set Are of Type 0

Let $G = (V, E)$ be a graph and $r \in \{0, 1, 2\}^V$ a vector of node types. Let $\{u, v\}$ be an articulation set with $r_u = r_v = 0$, so that $G - \{u, v\}$ has two components $\bar{G}_1 = (\bar{V}_1, \bar{E}_1)$ and $\bar{G}_2 = (\bar{V}_2, \bar{E}_2)$ with $r(\bar{V}_1) = 1$ and $r(\bar{V}_2) \geq 1$. We augment the component $\bar{G}_1$ to a graph $G_1 = (V_1, E_1)$ by adding the nodes u and v and all edges in E leading from u and v to the nodes in $\bar{G}_1$. In the same way, we construct a graph G_2 from $\bar{G}_2$. If there exist edges uv, we add them either to G_1 or to G_2, but not to both.

If $C \subseteq E$ is a feasible solution to **2ECON** on G, the set $T := C \cap E_1$ (called a *partial tree*) may have four different forms, according to whether u and v are connected in T, or whether u or v are used at all. More exactly, T is a feasible solution to one of the four following Steiner tree problems: $\mathbf{P_a}$, $\mathbf{P_b}$, $\mathbf{P_c}$, $\mathbf{P_d}$. These subproblems are defined on G_1 and use the same costs and node types for all nodes in $V_1 \backslash \{u, v\}$ as in the original problem; only the node types of u and v vary as follows:

$\mathbf{P_a}$: $r_u = 1$ and $r_v = 1$ and an artificial edge uv lpwith cost 0 is added;

$\mathbf{P_b}$: $r_u = 0$ and $r_v = 1$;

$\mathbf{P_c}$: $r_u = 1$ and $r_v = 0$;

$\mathbf{P_d}$: $r_u = 1$ and $r_v = 1$.

See Figure 2 for possible feasible solutions to these problems.

We say that a partial tree T is of type a if $T \cup \{uv\}$ is feasible for $\mathbf{P_a}$, and we say that T is of type b (respectively, c or d), if T is feasible for $\mathbf{P_b}$ (respectively, $\mathbf{P_c}$ or $\mathbf{P_d}$). Clearly, a partial tree of type d is also feasible for the network design problems $\mathbf{P_a}$, $\mathbf{P_b}$, and $\mathbf{P_c}$, but generally a solution of type a is not feasible for the network design problem $\mathbf{P_d}$. So the set of feasible solutions of type d is a subset of the feasible solutions of type a.

Let T_a, T_b, T_c, T_d be the four optimal solutions of each type a, b, c, d with values a, b, c, d, respectively. Note that $d \geq \max\{a, b, c\}$.

Now we replace the graph G_1 in G by a simpler graph $G'_1 = (V'_1, E'_1)$ (the *gadget*), consisting of three edges uu', $u'v'$, $v'v$ with edge weights depending on the values a, b, c, d (see Figure 3). The nodes u' and v' receive node type 1; u and v retain their node types. We call the resulting graph G' (= G_2 plus the gadget).

For any feasible solution C' in G', the set $C' \cap E'_1$ has only four different forms, namely,

$T'_a := E'_1 \backslash \{u'v'\}$,

$T'_b := E'_1 \backslash \{uu'\}$,

$T'_c := E'_1 \backslash \{vv'\}$,

$T'_d := E'_1$.

Let k denote the value $a + b + c - 2d$.

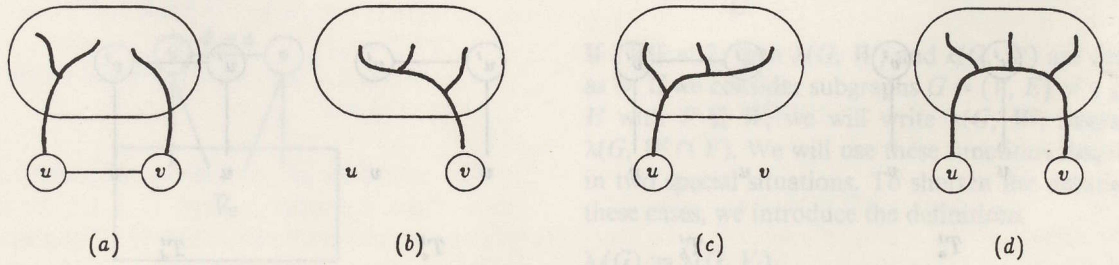

Figure 2. Partial trees.

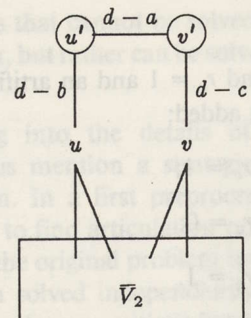

Figure 3. Gadget.

Clearly, if C' is feasible for the **2ECON** problem defined on G' and the set $T' := C' \cap E_1'$ is T_a', then T_a' can be replaced in C' by T_a to get a feasible solution for the **2ECON** problem on G, and conversely. Moreover, the edge weights in the gadget and the constant k are chosen so that $C(T_a) = c'(T_a') + k$.

The graph G' with gadget G_1' and edge weights is displayed in Figure 3, and the four possible partial solutions T_a', T_b', etc. are shown in Figure 4. Note that $d \geq \max\{a, b, c\}$, so the edge weights are nonnegative.

On graph G' exactly one **2ECON** problem is solved. Let C' be the optimal solution. C' can be written as $C_2 \cup T'$ with $C_2 \subseteq E_2$ and $T' \subseteq E_1'$. If $T' = T_a'$, then $C := C_2 \cup T_a$ is a feasible solution in G with the same cost as C' (except for the additive constant k) because $c(C') + k = c(C_2) + c'(T_a') + k = c(C_2) + c(T_a) = c(C)$.

Therefore, if C' was optimal for the **2ECON** problem on G', then C is optimal for the **2ECON** problem on G, and conversely.

Let us summarize this decomposition procedure.

Algorithm 1

We assume that G contains two nodes u, v of type 0, such that $G - \{u, v\}$ has two components $\overline{G}_1 = (\overline{V}_1, \overline{E}_1)$ and $\overline{G}_2 = (\overline{V}_2, \overline{E}_2)$ with $r(\overline{V}_1) = 1$ and $r(\overline{V}_2) \geq 1$.

Step 1. Construct two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ from $\overline{G}_1$ and $\overline{G}_2$, as described above.

Step 2. Solve the four Steiner tree problems $\mathbf{P_a}$, $\mathbf{P_b}$, $\mathbf{P_c}$, $\mathbf{P_d}$ in G_1. Let T_a, T_b, T_c, T_d be the corresponding optimal solutions with costs a, b, c, and d, respectively (see Figure 2).

Step 3. Construct graph G' (see Figure 3) and solve one **2ECON** problem (or the **2NCON** or Steiner-tree problems) on this graph. (Here Algorithm 1 may be called recursively.)

Step 4. Let C' be the optimal solution found in Step 3. Set $C_2 := C' \cap E_2$ and $T := C' \setminus C_2$.
If $T = T_a'$ set $C := C_2 \cup T_a$,
if $T = T_b'$ set $C := C_2 \cup T_b$, etc.

Step 5. C is the optimal solution for the **2ECON** problem on G (or the **2NCON** or the Steiner-tree problems).

2.3.2. Only One Node in the Articulation Set is of Type 0

Further simplification is possible if one of the cut nodes u and v, say u, is of type at least 1, and if $r_v = 0$. We can decompose graph G into two graphs G_1 and G_2, as above. Let the partial trees in G_1 be defined as above. In this case, it is not necessary to distinguish between partial trees of type a or b, so we can replace G_1 in G by a gadget G_1' with only two edges and an artificial node v'; r_u and r_v retain their former values, and $r_{v'}$ is set to 1 (see Figure 5). The values a, c and d are defined as the optimal values of the Steiner tree problems $\mathbf{P_a}$, $\mathbf{P_c}$, and $\mathbf{P_d}$ and the additive constant k is defined as $a + c - d$. We set $T_a' := \{v'v\}$, $T_c' := \{v'u\}$, and $T_d' := \{uv', v'v\}$. Now we can proceed with Step 3 of Algorithm 1 to find an optimal solution to the **2ECON** problem on G.

2.3.3. No Node in the Articulation Set is of Type 0

If both nodes u and v in the articulation set are of type at least 1, we need only distinguish between partial solutions of type a or d, so we compute the values a and d and solve the **2ECON** problem on

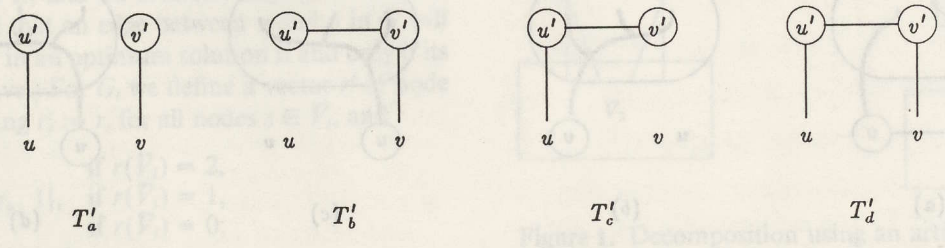

Figure 4. Types of solutions in the gadget.

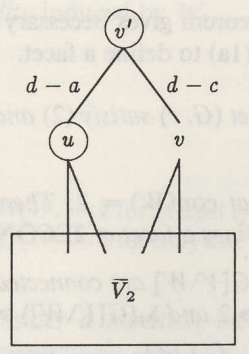

Figure 5. Gadget.

graph G_2 with an additional *uv*-edge of cost $d - a$. The additive constant k is a (see Figure 6). If the edge *uv* is used by an optimal solution C, we replace *uv* in C by the optimal partial tree of type d, else we augment C by the optimal partial tree of type a.

This is easy to implement in the special case (and was implemented by us) that some node w of type 1 has exactly two neighbor nodes u and v, both of type at least 1. In this case, a is $\min\{c(uw), c(wv)\}$ and $d - a$ is $\max\{c(uw), c(wv)\}$. The edge with the lower cost is always used in an optimal solution, so we can contract it and keep its weight, namely a, as an additive constant. The **2ECON** problem can then be solved on the contracted graph. This is exactly what the "decomposition" amounts to in this simple case. All other decomposition techniques mentioned in subsection 2.3 have not yet been implemented.

We also implemented a rather trivial reduction for nodes of type 0 and degree 2. In this case, if all costs are nonnegative, an optimal solution will either use both edges or none. So one edge may be shrunk and its cost added to the other edge.

We have devised, but not implemented, a further decomposition procedure for **2ECON** using a cut $S \subseteq E$ consisting of three edges so that in $G - S$ two nodes of type 2 are disconnected. Here ten types of "partial solutions" are needed. We do not want to

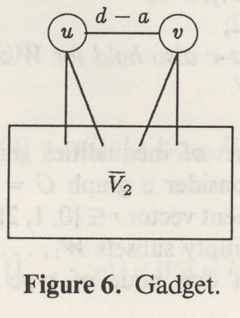

Figure 6. Gadget.

discuss the details of this decomposition here. An area of future research is decompositions on articulation sets of size 3 separating two nodes of type 2. This may become very complicated because many types of "partial solutions" have to be considered.

Let us give a short summary of the situations that are recognized and exploited by our code in a first preprocessing stage:

1. bridges and articulation nodes disconnecting two nodes of type at least 1; this includes an infeasibility test for **2NCON** (respectively, **2ECON**) if there exists an articulation node (respectively, bridge) disconnecting two nodes of type 2;
2. nodes of degree 2.

For finding bridges and articulation nodes we use a simple depth-first search algorithm described by Hopcroft and Tarjan (1973a). For finding articulation sets of size 2 there also exists a linear time algorithm (see Hopcroft and Tarjan 1973b), based on depth-first search, but that has not been implemented. Even so, using the (trivial) decompositions listed above on the real-world problems that we encountered, the graph sizes could be reduced by about one-half on the average.

3. FACET-INDUCING INEQUALITIES

In this section, we describe several classes of facet-inducing inequalities for the **2ECON** and **2NCON** polytopes, which were derived in Grötschel and Monma, and Grötschel, Monma and Stoer (1989).

Let $G = (V, E)$ be a graph and let $W \subseteq V$ with $|W| \geq 2$. We set

$\lambda(G, W) :=$ the minimum cardinality of a subset of E, whose removal from G disconnects two nodes of W, and

$\kappa(G, W) :=$ the minimum cardinality of a set $S \subseteq V \cup E'$ whose removal from G' disconnects two nodes of W, where $G' = (V, E')$ is the simple graph underlying G.

If $|W| < 2$, then $\lambda(G, W)$ and $\kappa(G, W)$ are defined as ∞. If we consider subgraphs $G = (V, E)$ of a graph H with $V \subseteq W$, we will write $\lambda(G, W)$ instead of $\lambda(G, W \cap V)$. We will use these functions frequently in two special situations. To shorten the notation in these cases, we introduce the definitions

$\lambda_i(G) := \lambda(G, V_i)$,

$\kappa_i(G) := \kappa(G, V_i)$,

where $V_i := \{v \in V \mid r_v \geq i\}$, $i = 0, 1, 2$. So $\lambda_0(G)$ is nothing but the edge connectivity of G, and $\kappa_0(G)$ is the node connectivity of G.

Throughout this section we make the following assumptions on (G, r).

a. $r \in \{0, 1, 2\}^V$ and at least two nodes $s \neq t$ satisfy $r_s = r_t = 2$;
b. if we consider the **2ECON** problem, we assume that G is 2-node connected and $\lambda_2(G) \geq 3$;
c. if we consider the **2NCON** problem, we assume that G is 2-node connected and $\kappa_2(G) \geq 3$. (2)

We will say that (G, r) satisfies (2) and mean that the graph $G = (V, E)$ and the vector $r \in \mathbb{Z}_+^V$ of connectivity types satisfy conditions (2) a, b, and c. If (2a) is not satisfied, the **2ECON** problem becomes the Steiner-tree problem. We do not want to consider this special case here. If b or c is not satisfied for some **2ECON** or **2NCON** problem, it is possible to decompose it into subproblems that satisfy b or c using the techniques explained in Section 2. For problem instances (G, r) satisfying (2), we know that the associated **2ECON** (respectively, **2NCON**) polyhedron is fully dimensional (Grötschel and Monma 1990, and Grötschel, Monma and Stoer 1989).

A full description of the necessary and sufficient conditions under which the inequalities listed in the sequel are facet-inducing is given in Grötschel, Monma and Stoer (1989). These conditions are quite complicated. Therefore, we will only give such necessary conditions, for each type of inequality, that helped to design and could be exploited by a separation routine.

An inequality $a^T x \leq a$ is *valid* with respect to a polyhedron P if $P \subseteq \{x \mid a^T x \leq \alpha\}$; the set $F_a := \{x \in P \mid a^T x = \alpha\}$ is called the *face* of P defined by $a^T x \leq \alpha$. If $\dim(F_a) = \dim(P) - 1$ and $F_a \neq \phi$, then F_a is a *facet* of P and $a^T x \leq \alpha$ is called *facet-defining* or *facet-inducing*.

Theorem 1 follows from Theorem 3.3 in Grötschel and Monma and characterizes which trivial inequalities (1c) define a facet.

Theorem 1. *Let (G, r) satisfy (2).*

a. *$x_e \leq 1$ defines a facet of* **2ECON**$(G; r)$ *and of* **2NCON**$(G; r)$ *for all $e \in E$.*
b. *$x_e \geq 0$ defines a facet of* **2ECON**$(G; r)$ *(respectively,* **2NCON**(G, r)*) for $e \in E$, if and only if for every edge $f \neq e$, the polytope* **2ECON**$(G - \{e, f\}; r)$ *(respectively,* **2NCON**$(G - \{e, f\}; r)$*) is nonempty.*

The next theorem gives necessary conditions for a *cut inequality* (1a) to define a facet.

Theorem 2. *Let (G, r) satisfy (2) and let $W \subseteq V$ with $\phi \neq W \neq V$.*

a. *Suppose that $\text{con}(W) = 2$. Then $x(\delta(W)) \geq 2 = \text{con}(W)$ defines a facet of* **2ECON**$(G; r)$ *only if*

 $G[W]$ and $G[V \setminus W]$ are connected; and
 $\lambda_1(G[W]) \geq 2$ and $\lambda_1(G[V \setminus W]) \geq 2$.

b. *Suppose that $\text{con}(W) = 1$. Then $x(\delta(W)) \geq 1 = \text{con}(W)$ defines a facet of* **2ECON**$((G; r)$ *if and only if*

 $G[W]$ and $G[V \setminus W]$ are connected;
 $\lambda_1(G[W]) \geq 2$ and $\lambda_1(G[V \setminus W]) \geq 2$;
 $\lambda_2(G[V \setminus W]) \geq 3$.

c. *Suppose that $\text{con}(W) = 0$. Then $x(\delta(W)) \geq 0 = \text{con}(W)$ does not define a facet of* **2ECON**$(G; r)$ *or of* **2NCON**$(G; r)$.

d. *Suppose that $\text{con}(W) = 2$. Then $x(\delta(W)) \geq 2$ defines a facet of* **2NCON**$(G; r)$ *only if*

 the conditions of a are satisfied; and
 $\kappa_2(G[W]) \geq 2$ and $\kappa_2(G[V \setminus W]) \geq 2$.

e. *Suppose that $\text{con}(W) = 1$. Then $x(\delta(W)) \geq 1$ defines a facet of* **2NCON**$(G; r)$ *only if*

 the conditions of b are satisfied; and
 $\kappa_2(G[V \setminus W] - e) \geq 2$ for all $e \in E(V \setminus W)$.

The following theorem gives some conditions for when the *node cut inequalities* (1b) do not define facets for **2NCON**$(G; r)$.

Theorem 3. *Let (G, r) satisfy (2) and let a node $z \in V$ and a set $W \subseteq V \setminus \{z\}$, $\phi \neq W \neq V \setminus \{z\}$ with $r(W) = 2$ and $r(V \setminus (W \cup \{z\})) = 2$ be given. The node cut inequality $x(\delta_{G-z}(W)) \geq 1$ defines a facet of* **2NCON**$(G; r)$ *only if:*

a. *$G[W]$ is connected;*
b. *$\lambda_1(G[W \cup \{z\}]) \geq 2$;*
c. *$\lambda_2 G[W]) \geq 2$;*
d. *Conditions a–c also hold for $\bar{W} := V \setminus (W \cup \{z\})$ instead of W.*

The next class of inequalities generalizes the cut inequalities. Consider a graph $G = (V, E)$ together with a requirement vector $r \in \{0, 1, 2\}^V$ and a partition of V into nonempty subsets $W_1, \ldots, W_p$, each containing at least one node of type at least 1. The *partition inequality* induced by $W_1, \ldots, W_p$ is given by

$$\frac{1}{2}\sum_{i=1}^{p} x(\delta(W_i)) \geqslant \begin{cases} p & \text{if } r(W_i)=2 \text{ for at least two sets } W_i, \\ p-1 & \text{if } r(W_i)=2 \text{ for only one set } W_i. \end{cases} \quad (3)$$

Theorem 4. *Consider a partition inequality* (3) *for* **2ECON**$(G; r)$ *(respectively,* **2NCON** $(G; r)$*). Denote by $\hat{G}$ the graph obtained by shrinking node sets W_i in G to nodes w_i of type $\hat{r}(w_i) := r(W_i)$.*

a. *Suppose that $r(W_i) = 2$ for at least two sets W_i. The partition inequality defines a facet of* **2NCON**$(G; r)$ *only if*
   - $\kappa_2(\hat{G}) \geqslant 3$ *and* $\kappa_1(\hat{G}) \geqslant 2$;
   - *in $\hat{G}$ every node of type 2 is adjacent to some node of type 1;*
   - *$\hat{G}$ has a cycle C containing all nodes of type 2;*
   - *$G[W_i]$ is connected for $i = 1, \ldots, p$;*
   - $\lambda_1(G[W_i]) \geqslant 2$ *for* $i = 1, \ldots, p$.

b. *Suppose that $r(W_i) = 2$ for only one set W_i. The partition inequality defines a facet of* **2NCON**$(G; r)$ *only if*
   - $\kappa_1(\hat{G}) \geqslant 2$;
   - *$G[W_i]$ is connected for $i = 1, \ldots, p$;*
   - $\lambda_1(G[W_i]) \geqslant 2$ *for* $i = 1, \ldots, p$;
   - $\lambda_2(G[W_i]) \geqslant 3$ *for the set W_i with $r(W_i) = 2$.*

The class of node cut inequalities can be generalized in a similar way as the class of cut inequalities. The following class of node partition inequalities are valid for the **2NCON** polytope, but are not generally valid for the **2ECON** polytope. Let $G = (V, E)$ be a graph and $r \in \{0, 1, 2\}^V$. Let $z \in V$ and let $W_1, \ldots, W_p$ be a partition of $V \setminus \{z\}$ into nonempty node sets W_i with $r(W_i) = 2$ for at least two node sets. The following *node partition inequality* induced by z and $W_1, \ldots, W_p$ is given by

$$\frac{1}{2}\left(\sum_{i\in I_2} x(\delta_{G-z}(W_i)) + \sum_{i\in I_1} x(\delta_G(W_i)) + x([\{z\} : \cup_{i\in I_1} W_i])\right) \geqslant p-1, \quad (4)$$

where $I_k := \{i \in \{1, \ldots, p\} \mid r(W_i) = k\}$, $k = 1, 2$.

Theorem 5. *The node partition inequality* (4) *defines a facet of* **2NCON**$(G; r)$ *only if*
- *$G[W_i]$ is connected for all $i \in I_1$;*
- $\lambda_1(G[W_i \cup \{z\}]) \geqslant 2$ *for all* $i \in I_2$;
- $\lambda_1(G[W_i]) \geqslant 2$ *for all* $i \in I_1$;
- $\lambda_2(G[W_i]) \geqslant 2$ *for* $i = 1, \ldots, p$.

The next class of inequalities is closely related to the 2-matching inequalities for the traveling salesman problem (see Grötschel and Padberg 1985). A subclass of it is also valid for the polytope of *2-covers* of G (i.e., those subgraphs of G where each node has a degree of at least 2).

Consider a subset $H \subseteq V$ called the *handle* and a subset $T \subseteq \delta(H)$. For each $e \in T$ we denote by T_e the set of the two endnodes of e. The sets T_e, $e \in T$, are called *teeth*. For simplicity, we also call the edges $e \in T$ teeth in this paper. Furthermore, H is partitioned into $p \geqslant 3$ sets $H_1, H_2, \ldots, H_p$, with
- $r(H_i) \geqslant 1$ for $i = 1, \ldots, p$;
- $r(H_i) = 2$ if H_i has a nonempty intersection with some tooth, $i = 1, \ldots, p$;
- no more than two teeth may intersect any H_i, $i = 1, \ldots, p$;
- $|T| \geqslant 3$ and odd.

The *lifted 2-cover inequality* induced by T and $H_1, \ldots, H_p$ is defined by

$$x(E(H)) - \sum_{i=1}^{p} x(E(H_i)) + x(\delta(H)) - x(T) \geqslant p - \left\lfloor \frac{|T|}{2} \right\rfloor, \quad (5)$$

where $\lfloor x \rfloor$ denotes the largest integer not greater than x. It is valid for **2ECON**$(G; r)$, but it does not generally define a facet of **2NCON**$(G; r)$.

Theorem 6. *Let $H_1, \ldots, H_p$ and T that satisfy the requirements above induce a lifted 2-cover inequality. Denote by $\hat{G}$ the graph obtained from G by contracting each set H_i to a vertex h_i of connectivity type $r(H_i)$. A lifted 2-cover inequality* (5) *defines a facet of* **2ECON**$(G; r)$ *only if*
- *$\hat{G}[H]$ is connected;*
- $\lambda_2(\hat{G}[H]) \geqslant 2$;
- *$G[H_i]$ is connected for $i = 1, \ldots, p$;*
- $\lambda_1(G[H_i]) \geqslant 2$ *for* $i = 1, \ldots, p$.

If all nodes in V are supposed to be of type 2, and if we take the H_i to consist of only one node each, we arrive at a subclass of the lifted 2-cover inequalities, the so-called *2-cover constraints:*

$$x(E(H)) + x(\delta(H)\setminus T) \geqslant |H| - \lfloor |T|/2 \rfloor \quad \text{for all } H \subseteq V \text{ and all } T \subseteq \delta(H). \quad (6)$$

These inequalities are valid for the polytope of 2-covers. In fact, a complete description for the polytope of 2-covers—consisting of the degree constraints in (7), the 2-cover constraints (6), and the trivial constraints (1c)—can be derived from a complete description of the b-matching polytope by viewing the complement $(V, E\setminus C)$ of a 2-cover (V, C) as a subgraph of (V, E), whose node degrees may not exceed a given number b_v. The analogy between 2-covers and b-matchings allows us to use an algorithm from b-matching theory to treat lifted 2-cover inequalities, as will be seen in the next section. More theoretical details can be found in Cook and Pulleyblank (1987), and Grötschel, Monma and Stoer (1991).

There are more classes of inequalities valid for **2ECON**$(G; r)$ or **2NCON**$(G; r)$ known; see, for instance, Grötschel, Monma and Stoer (1989). But in the design of our cutting plane algorithm we restricted attention to the classes mentioned above.

4. IMPLEMENTATION OF THE CUTTING PLANE ALGORITHM

In this section, we give an outline of the cutting plane algorithm for solving **2ECON** problems, and we will describe our separation routines for partition, node partition, and lifted 2-cover inequalities.

4.1. An Outline of the Cutting Plane Procedure

In a first preprocessing stage we identify the so-called *essential edges* (edges that have to be used by any feasible solution), and we decompose the problem into several nondecomposable subproblems that can be solved independently of each other. The tools for this procedure are explained in Section 2. After having solved all subproblems, we can put the solutions together in a straightforward manner.

Let us assume therefore that we are given a **2ECON** problem on (G, r) that cannot be decomposed into independent subproblems, i.e., (G, r) satisfies (2). If nothing else is said, the statements made in this subsection for **2ECON** also apply to **2NCON**.

The cutting plane procedure starts with solving the LP

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{subject to} \\ &x(\delta(v)) \geqslant r_v \quad \text{for all } v \in V \text{ with } r_v \geqslant 1, \\ &0 \leqslant x_e \leqslant 1 \quad \text{for all } e \in E \end{aligned} \quad (7)$$

consisting of at most $|V|$ degree inequalities and $2|E|$ trivial inequalities. Almost all of these define facets (see Theorems 1 and 2a and b) of **2ECON**$(G; r)$.

The optimal solution $y \in \mathbb{R}^E$ of this relaxation of **2ECON** is usually not feasible for the polytope **2ECON**$(G; r)$. (If it were, we would be finished.) So, in each iteration of the cutting plane algorithm we try to find inequalities (more specifically: partition and lifted 2-cover inequalities) that are valid for **2ECON**$(G; r)$, but are violated by y. (In the case of a **2NCON** problem, we also try to find violated node partition inequalities.) Geometrically, such an inequality defines a hyperplane in $\mathbb{R}^E$ separating y from the **2ECON** polyhedron, a so-called "cutting plane." The exact algorithms and the heuristics for finding violated inequalities violated by a given y are called *separation routines.*

We add all the violated inequalities found by our separation routines to the current LP and solve the revised LP to get a new optimum solution y. (We do not solve the new LP from scratch, but use postoptimization.) We repeat this process until the current optimal LP solution y happens to be feasible for **2ECON**$(G; r)$, or no further partition or lifted 2-cover inequalities violated by y are found. In the second case, we proceed with a branch-and-cut method. This enumerative technique had to be applied in only three of our real-world examples.

In the first case (y is feasible) we know that y is optimal because the present LP is a relaxation of the **2ECON** problem. Note that the feasibility of y is identical with y being a $\{0, 1\}$-vector that satisfies all cut constraints (1a). This feasibility criterion is easy to check.

Of course, since we are using only a subset of all facet-defining inequalities for **2ECON** $(G; r)$, we cannot be sure to find an optimal solution with such a cutting plane algorithm for *all* graphs G, cost functions c, and connectivity types r. For the majority of the real-world examples known to us, the relaxation of the **2ECON** polyhedron using partition and lifted 2-cover inequalities was sufficient to find the optimal solution.

In any case, even if the present fractional solution y is not feasible, its objective function value $c^T y$ provides a lower bound for the **2ECON** problem, which is increased with every iteration (or at least, it does not drop). With these lower bounds we can show that the heuristic methods of Monma and Shallcross for the **2ECON** problem perform very well (see Section 5).

We summarize the cutting plane algorithm next.

Cutting Plane Algorithm for the 2ECON (or 2NCON) Problem

Step 1. Decompose the **2ECON** (or **2NCON**) problem given by (G, r) into independent subproblems.

Step 2. For each subproblem do Steps 3–5:

Step 3. Solve the LP (7). Let y be the optimal solution to this LP.

Step 4. While y is not feasible for **2ECON**$(G; r)$ (or **2NCON**$(G; r)$) do Step 5:

Step 5. Find violated partition and lifted 2-cover inequalities, (in the case of **2NCON**, also node partition inequalities), add them to the LP, and solve it.

Let y be the new LP solution. If no violated inequalities can be found, branch on some variable with fractional value.

Step 6. Put the solutions of all the subproblems together.

In Sections 4.3–4.8 we describe the separation routines used in Step 5. In Section 4.2 we show that the separation problems associated with partition inequalities, node partition inequalities, and lifted 2-cover inequalities are NP-hard. This means that our separation routines for these types of inequalities can only be heuristics.

4.2. Complexity of the Separation Problems

In this section, we show that the separation problems associated with partition, node partition, and lifted 2-cover inequalities are NP-hard. The same is true for the class of partition inequalities for the Steiner-tree polytope (see Grötschel and Monma). Note that the exact separation for the cut inequalities (1a) and the node cut inequalities (1b) can be performed in polynomial time using the Gomory-Hu algorithm.

We start with the partition inequalities for the Steiner-tree problem. Let $G = (V, E)$ be a graph with connectivity types $r_v \in \{0, 1\}$ for all $v \in V$. The *Steiner-tree polytope* is defined as the convex hull of all incidence vectors X^F, where F contains a Steiner tree of G spanning the set of "terminal nodes" $\{v \in V : r_v = 1\}$. It can also be characterized as the set of $\{0, 1\}$-vectors satisfying inequalities (1a, c and d).

Let $W_1, \ldots, W_p$ be a partition of V into p nonempty node sets each containing at least one node of type 1. The partition inequality

$$\frac{1}{2}\sum_{i=1}^{p} x(\delta(W_i)) \geqslant p - 1, \qquad (8)$$

is clearly valid for the Steiner-tree polytope (see also Grötschel and Monma).

The separation problem for this and other classes of inequalities is defined as follows.

The separation problem for a given class of inequalities. *Given a graph $G = (V, E)$, connectivity types $r_v \in \{0, 1, 2\}$ for all $v \in V$, and values $y_e \geqslant 0$ for all $e \in E$. Decide whether y satisfies all inequalities of the given class and, if not, find an inequality of the given class violated by y.*

In our NP-hardness proof, we make use of an optimization problem related to the separation problem, with the difference that we want to determine a "most violated" inequality. Let us first introduce an additional notion.

RELSEP for a given class of inequalities. *Given a graph $G = (V, E)$, connectivity types $r_v \in \{0, 1, 2\}$ for all $v \in V$, and values $y_e \geqslant 0$ for all $e \in E$. (The inequalities in the given class are supposed to be of the form $a^T x \geqslant b$ with $b > 0$.) Decide whether y satisfies all inequalities of the given class and, if not, find an inequality $a^T x \geqslant b$ of the class such that $(b - a^T y)/b$ is maximized.*

We first show NP-hardness of **RELSEP** for our various classes of inequalities by reduction from the 3-way cut problem. NP-hardness of the 3-way cut problem was shown, or rather indicated, in an extended abstract of Dahlhaus et al. (1983).

For a graph $G = (V, E)$ and three different nodes s_1, s_2, s_3 (called *special nodes*), a *3-way cut* is an edge set $C \subseteq E$, such that s_1, s_2, s_3 are in different components of $G - C$. If $W_1, W_2, W_3 \subseteq V$ are a partition of V, such that $s_i \in W_i$ and $s_k \notin W_i$ for $i, k \in \{1, 2, 3\}$, $i \neq k$, then $\bigcup_{i=1}^{3} \delta(W_i)$ is a 3-way cut; moreover, every 3-way cut clearly contains such a special 3-way cut.

The problem of determining a minimum 3-way cut in a graph. *Given a graph $G = (V, E)$ and vertices $s_1, s_2, s_3 \in V$, determine a 3-way cut $C \subseteq E$ of minimum cardinality.*

Theorem 7. RELSEP *for partition inequalities* (8) *is NP-hard.*

Proof. Suppose that we have an instance of the 3-way cut problem given by $G = (V, E)$ and three special nodes s_1, s_2, and s_3. We transform this instance into an instance of **RELSEP** for partition inequalities. First we use a standard min-cut algorithm to determine, for $i = 1, 2, 3$, a cut $\delta(S_i)$ of minimum cardinality with $s_i \in S_i$ that separates s_i from the other two special nodes. We may assume that $|\delta(S_1)| \leqslant |\delta(S_2)| \leqslant |\delta(S_3)|$. We set $r(s_1) = r(s_2) = r(s_3) := 1$ and $r_v := 0$ for all other nodes v and define $y'_e := 1$ for all $e \in E$. Note that finding a 3-way cut of minimum cardinality in G is equivalent to finding a partition inequality (8) with $p = 3$, such that $\frac{1}{2}\sum_{i=1}^{3} y'(\delta(W_i))$ is as small as possible.

Next, we construct a graph G' from G by adding two edges s_1s_2 and s_1s_3 with $y'(s_1s_2) = y'(s_1s_3) := |\delta(S_2)| - |\delta(S_1)| + 1$ ($=: k$). With the addition of these edges, the left-hand side of a partition inequality (8) with $p = 3$, namely $\frac{1}{2}\sum_{i=1}^{3} y'(W_i)$, is changed only by the constant $2k$. The number k is chosen so that $\delta(S_2)$ is the minimum cut in G' (with respect to the capacities y') disconnecting any two special nodes.

We now call the algorithm for **RELSEP** with input G', r, and $y := y'/1^T y'$. By our definition, y violates all partition inequalities. So the algorithm for **RELSEP** will output one. If it is a partition inequality (8) with $p = 3$, then clearly the three sets of the partition determine a minimum cardinality 3-way cut. If it is a partition inequality with $p = 2$, we know that for a minimum 3-way cut of cardinality, say c (in G),

$$\frac{c + 2k}{2} \geqslant |\delta_G(S_2)| + k.$$

This implies that $c \geqslant 2|\delta(S_2)| \geqslant |\delta(S_1) \cup \delta(S_2)|$. Since $\delta(S_1) \cup \delta(S_2)$ is also a 3-way cut in G, it must be a minimum one. In either case, we have found a minimum 3-way cut of G. So **RELSEP** for partition inequalities (8) is NP-hard.

NP-hardness of **RELSEP** for lifted 2-cover and node partition inequalities can be shown in a similar way.

Theorem 8. RELSEP *for node partition inequalities* (4), *lifted 2-cover inequalities* (5), *and partition inequalities* (3) *is NP-hard.*

The following result shows that with an exact separation algorithm, for any class of inequalities with certain properties, we can solve (in polynomial time) the associated **RELSEP** problem. The proof uses the polynomial-time equivalence relationships between certain algorithmic problems established in Grötschel, Lovász and Schrijver (1988).

Theorem 9. *Let $a_i^T x \geqslant b_i$, $i = 1, \ldots, m$, be a system of inequalities in $\mathbb{R}^n$ with $b_i > 0$ for $i = 1, \ldots, m$. Then the problem of separating nonnegative vectors from this class of inequalities is polynomial-time equivalent to* **RELSEP** *for the same class.*

Proof. Without loss of generality, we may assume that the class of inequalities is given by $a_i^T x \geqslant 1$ for $i = 1, \ldots, m$. Let e_i, $i = 1, \ldots, n$, be the unit vectors in $\mathbb{R}^n$ with the ith component equal to one and all other components equal to zero. Given $y \geqslant 0$, the answer to **RELSEP** can be found by solving

minimize $u^T y$

subject to

$u \in P' := \text{conv}\{a_1, \ldots, a_m\} + \text{cone}\{e_1, \ldots, e_n\}$.

Note that this optimization problem can be solved using the ellipsoid method (see Grötschel, Lovász and Schrijver) if there is an oracle testing membership for P'. It is easy to see that $u \in P'$ if and only if $u^T x \geqslant 1$ is valid for the polyhedron $P := \{x \mid a_i^T x \geqslant 1, i = 1 \ldots, m; x \geqslant 0\}$. Finally, the validity problem for P can be reduced to separation over P, again by using a result from Grötschel, Lovász and Schrijver. Therefore, a polynomial algorithm for the problem of separating nonnegative vectors can be used to solve **RELSEP** in polynomial time. The reverse direction is trivially true.

This proves the following corollary.

Corollary. *The separation problems associated with the classes of partition inequalities* (8) *for the Steiner-tree polytope, partition inequalities* (3), *node partition inequalities* (4), *and lifted 2-cover inequalities* (5) *are NP-hard.*

4.3. Heuristics for Separating Partition Inequalities

Let (G, r) satisfy (2), and let y be some point in $\mathbb{R}^E$ with $0 \leqslant y_e \leqslant 1$. Our aim is to find a partition inequality violated by this point. By the results of Section 4.2, it seems hopeless to find an efficient exact algorithm for the separation of partition inequalities, therefore, we have to use heuristics. Nevertheless, it is possible to solve the separation problem for the cut constraints (1a) in polynomial time. So, in our heuristics we often use "almost violated" (cut) inequalities and transform them into violated partition inequalities. Here, an *almost violated* inequality is an inequality $a^Tx \geqslant b$ with $a^Ty \leqslant b + \alpha$ for some "small" parameter α (we used $\alpha = 0.5$); $a^Tx \geqslant b$ is a *violated* inequality; if $a^Ty < b$.

The heuristic that we applied has the following general form.

Heuristic 1. Finding Violated Partition Inequalities

Step 1. Shrink all or some edges $e \in E$ with the property that any violated partition inequality using this edge can be transformed into some at least as violated partition inequality not using this edge. ("Using e" means: e has a coefficient of 1 in the partition inequality.)

Step 2. Find some violated or almost violated cut constraints in the resulting graph.

Step 3. Attempt to modify these cut constraints into violated partition inequalities.

We will explain our shrinking criteria of Step 1 in Section 4.4. Here we will specify Steps 2 and 3 of the general heuristic.

In Step 2 we apply the algorithm of Gomory-Hu (1961) to graph G with costs y to find a cut $\delta(W)$ with minimum cost $y(\delta(W))$. This algorithm produces the so-called Gomory-Hu tree with the property that for all pairs s, t of nodes the minimum $[s, t]$-cut in the tree is also a minimum $[s, t]$-cut in G. Gusfield (1987) described a version of this algorithm that is very easy to implement. It consists of solving $|V| - 1$ maximum flow problems on the graph G in a certain order. As with our max-flow routine we used the algorithm of Goldberg and Tarjan (1987) in an implementation we obtained from R. Ahuja. This code is the fastest max-flow algorithm available to us.

In Step 3 we are given some violated or almost violated cut constraint $x(\delta(W)) \geqslant \text{con}(W)$ derived from the Gomory-Hu tree.

In many cases, the given cut inequality does not define a facet because $G[W]$ contains a bridge e (a so-called *Steiner bridge*), so that $G[W] - e$ has two components with node sets W_1 and W_2 both containing nodes of type 1 or 2; i.e., the condition $\lambda_1(G[W]) \geqslant 2$ of Theorem 2a or b is violated. In such a case, the cut constraint $x(\delta(W)) \geqslant 2$ can be written as the sum of a partition inequality with partition $\{W_1, W_2, V\backslash W\}$ and the trivial inequality $-x_e \geqslant -1$ (see Figure 7).

If the cut constraint is violated, the partition inequality is also violated. So, instead of adding the cut constraint to the LP, we add the partition inequality, which defines a face of higher dimension than the original cut constraint.

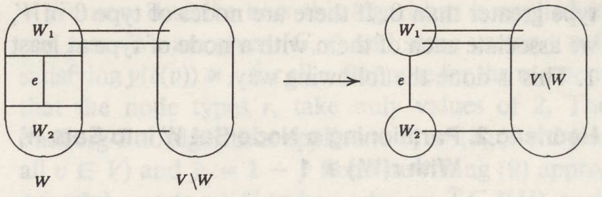

Figure 7. Splitting a cut into a partition.

There is also a practical reason for doing this: Using facets as cutting planes usually results in a more radical change in the structure of the fractional solution than using lower-dimensional faces. This is illustrated by the following example, which actually occurred in our computations. (See Figure 8, where solid lines denote $y_e = 1$, and dashed lines denote $y_e = \frac{1}{2}$.) Here, the point y shown on the left violates the cut constraint $x(\delta(\{u, v\})) \geqslant 1$, which is implied by the partition inequality with partition $\{\{u\}, \{v\}, V\backslash\{u, v\}\}$. When the cut inequality was added to the LP, the point y on the upper right was produced (satisfying the cut inequality but not the partition inequality), but when the partition inequality was added instead of the cut inequality, the point y on the lower right was produced, which looks (locally) feasible.

Also, the LP value changes more dramatically when we tested our cut constraints for Steiner bridges. For instance, in one problem, we obtained a lower bound of 1,419.65 using only cut constraints. After this, no more violated cuts were found. Using cut constraints *and* testing for Steiner bridges made the lower bound jump up to 1,474; the optimal value was 1,489. So facet characterizations do have some practical implications.

Another way of transforming a cut inequality $x(\delta(W)) \geqslant \text{con}(W)$ into a partition inequality is to split W into separate node sets $\{w\}$ for each $w \in W$. This requires that all nodes in W have a connectivity

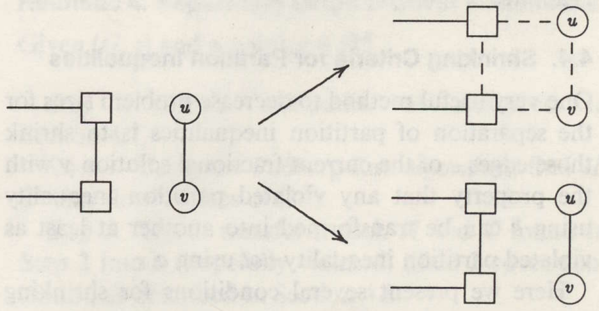

Figure 8. Using the partition instead of the cut inequality.

type greater than 0. If there are nodes of type 0 in W, we associate each of them with a node of type at least 1. This is done the following way.

Heuristic 2. Partitioning a Node Set W Into Sets W_i With $r(W_i) \geq 1$

Given (G, r), a node set W, and a point $y \in \mathbb{R}^E$.

Step 1. Set $G' := G[W]$ and $W' := W$.

Step 2. As long as there are nodes v of type 0 in W' do:
Find a neighbor w of v in G' with $w \in W$ and y_{vw} as large as possible.
Shrink edge vw to node w with connectivity type r_w.
Identify parallel edges uw (by adding their y values).

Step 3. For each $w_i \in W'$ set $W_i :=$ the subset of W represented by w_i.

A drawback of Heuristic 2 is that an inequality with many nonzero coefficients is created when W is large. So we use it only for "small" sets W, say, with not more than seven nodes of type at least 1.

Usually, our routines for splitting W work well only in the first few iterations of the cutting plane algorithm, because after a while the cuts produced in Step 2 of Heuristic 1 are only degree constraints of the form $x(\delta(w)) \geq r_w$.

So we also employ a heuristic that enlarges the shore of an (almost) violated cut $\delta(W)$ by adding a certain branch of the Gomory-Hu tree. That is, a node w is determined that is adjacent to W by an edge of "high" y_i value. The subtree (of the Gomory-Hu tree) defined by w is then added to set W. Now the splitting routines can be applied to W to derive some partition inequalities that hopefully are violated.

The Gomory-Hu algorithm (with about $O(n^4)$ worst case running time) may become very slow for large graphs because $|V| - 1$ max-flow problems have to be solved. In the next section, we explain how problem sizes can be reduced before the Gomory-Hu algorithm is applied.

4.4. Shrinking Criteria for Partition Inequalities

One very useful method to decrease problem sizes for the separation of partition inequalities is to shrink those edges e of the current fractional solution y with the property that any violated partition inequality using e can be transformed into another at least as violated partition inequality *not* using e.

Here we present several conditions for shrinking edges $e = uv$, depending on the value y_e, the connectivity types r_u and r_v, and the values $y(\delta(u))$ and $y(\delta(v))$. When shrinking edge uv, edges uw and vw are identified and the y-values are added. The connectivity type of the shrunk node is set to $\min\{r(V\setminus\{u, v\}), \max(r_u, r_v)\}$.

For each shrinking criterion, we give a proof of why there is a partition inequality not using e that is at least as violated as a partition inequality using e.

Criterion 1. $y_e \geq k := max(r_w : w \in V)$.

Proof. Suppose that $\{W_1, \ldots, W_p\}$ induces a violated partition inequality with $u \in W_1$ and $v \in W_2$. The partition inequality has a right-hand side of $p - 1$ if all nodes of type 2 are contained in one W_i. If $p \geq 3$, the sets $W_1 \cup W_2, W_3, \ldots, W_p$ also induce a partition inequality. The left-hand side decreases by at least $y_e \geq k$; the right-hand side decreases by at most k. (Note that if $k = 2$, it may happen that W_1 and W_2 are the only node sets in the partition containing nodes of type 2. In this case, the right-hand side of $\{W_1 \cup W_2, W_3, \ldots, W_p\}$ decreases by two.) So the new partition inequality is at least as violated as the original one, and it does not use edge e. If $p = 2$, then $\{W_1, \ldots, W_p\}$ does not define a violated partition inequality.

Criterion 2. $y_e \geq r_v$ *and* $y_e \geq y(\delta(v)) - y_e$.

Proof. Suppose that $\{W_1, W_2, \ldots, W_p\}$ induces a violated partition inequality with $u \in W_1$ and $v \in W_2$. If $r_v = 2$, Criterion 2 is the same as Criterion 1. If $r_v \leq 1$, and W_2 contains (besides v) some other node of type at least 1, we can replace $\{W_1, W_2, \ldots, W_p\}$ by $\{W_1 \cup \{v\}, W_2\setminus\{v\}, \ldots, W_p\}$. The difference in the left-hand side between the first and the second partition inequality is

$$y([\{v\} : W_1]) - y([\{v\} : W_2]) \geq y_e - (y(\delta(v)) - y_e) \geq 0.$$

The right-hand sides do not differ, as $W_1 \cup \{v\}$ and $W_2\setminus\{v\}$ still contain nodes of type 2. Therefore, the second partition inequality is at least as violated as the first, and it does not use e.

The above transformation does not work if $r_v = 1$ and W_2 contains (besides v) only nodes of type 0. But, the partition inequality induced by $\{W_1 \cup W_2, W_3, \ldots, W_p\}$ is at least as violated as the partition inequality induced by $\{W_1, W_2, \ldots, W_p\}$ if $p \geq 3$. If $p = 2$, the partition (or cut) inequality induced by $\{W_1, W_2\}$ cannot be violated, because it has a right-hand side of 1.

4.5. Separating Node Partition Inequalities

Let (G, r) be an instance of the **2NCON** problem and let y be some point in $\mathbb{R}^E$ with $0 \leq y_e \leq 1$ for all $e \in E$. Our aim is to find a node z and a partition

$\{W_1, \ldots, W_p\}$ of $V \backslash \{z\}$, such that the node partition inequality induced by z and $\{W_1, \ldots, W_p\}$ is violated by y. The candidate nodes z are determined as the articulation nodes of the graph $G' := (V, E')$, where E' contains all edges of a y-value of at least ½. Once this is done, a partition of $V \backslash \{z\}$ is found as follows, using principally the same ideas as for the separation of partition inequalities.

Heuristic 3. Separating Node Partition Inequalities

Given (G, r) and a point $y \in \mathbb{R}^E$.

Step 1. For all articulation nodes z of graph G' separating at least two nodes of type ≥ 2, do:

Step 2. Shrink all (or some) edges e of $G - z$ with the property that any violated node cut inequality using this edge can be transformed into some at least as violated node cut inequality not using this edge.

Step 3. In the resulting shrunk graph G'' (not containing z) finds some cuts $\delta_{G''}(W)$ with $y(\delta_{G''}(W)) \leq 1 + \alpha$, where $\alpha := ½$.

Step 4. Attempt to transform one shore (W or $V \backslash W$) of such a cut $\delta_{G''}(W)$ into violated node partition inequalities of G.

The shrinking in Step 2 is done by criteria analogous to the criteria used for partition inequalities described in Section 4.4.

The cuts in Step 3 are found by using the Gomory-Hu algorithm. The transformation of cuts $\delta(W)$ into violated node partition inequalities in Step 4 is done by splitting W exactly as in Heuristic 2. Here $V \backslash W$ is the shore containing the node v with the highest value $y(\delta_{G''}(v))$.

4.6. Separating Lifted 2-Cover Constraints

The separation problem for lifted 2-cover constraints (5) is NP-complete, but there exists an efficient (polynomial-time) algorithm for separating 2-cover constraints (6). This exact separation routine is a straightforward modification of the algorithm of Padberg and Rao (1982) for the separation of 1-capacitated b-matching constraints.

The Padberg-Rao algorithm achieves the following. Given positive integers b_v for all $v \in V$ and a vector $\bar{y} \in \mathbb{R}^E_+$ with $\bar{y}(\delta(v)) \leq b_v$ for all $v \in V$, it finds a node set $H \subseteq V$ and an edge set $\bar{T} \subseteq \delta(H)$ with $\sum_{v \in H} b_v + |\bar{T}|$ odd such that the value

$$\frac{1}{2}\left(\sum_{v \in H} b_v + |\bar{T}| - 1 \right) - \bar{y}(E(H)) - \bar{y}(\bar{T}) \qquad (9)$$

is minimal.

We now describe how this algorithm is applied to our case. We are given (G, r) and some vector $y \geq 0$ satisfying $y(\delta(v)) \geq r_v$ for all v. Suppose for the moment that the node types r_v take only values of 2. The Padberg-Rao algorithm applied to $b_v := |\delta(v)| - 2$ (for all $v \in V$) and $\bar{y} := \mathbf{1} - y$ finds (rewriting (9) appropriately) a node set H and an edge set $\bar{T} \subseteq \delta(H)$, such that $T := \delta(H) \backslash \bar{T}$ is of odd cardinality, and

$$y(E(H)) + y(\delta(H) \backslash T) - |H| + (|T| - 1)/2$$

is minimal. Comparing this with the 2-cover inequality (6) we see that in the case when all node types are 2, the Padberg-Rao algorithm can be used to separate 2-cover inequalities exactly.

If the node types are 0, 1, and 2, the Padberg-Rao algorithm is applied to $b_v := |\delta(v)| - r_v$ for all nodes v of types 0 and 2, $b_v := |\delta(v)| - 2$ for all nodes v of type 1, and $\bar{y} := \mathbf{1} - y$. This setting may violate one of the necessary conditions for the application of the Padberg-Rao algorithm, namely $\bar{y}(\delta(v)) \leq b_v$ for all $v \in V$. We ignore this and use the Padberg-Rao algorithm to get a node set H and an odd-cardinality edge set $T := \delta(H) \backslash \bar{T}$ such that

$$y(E(H)) + y(\delta(H) \backslash T) - |H \cap V_1| + (|T| - 1)/2 \qquad (10)$$

is of (hopefully) low value, where $V_1 := \{v \in V \mid r_v \geq 1\}$. This expression may not correspond to a lifted 2-cover inequality (5) if H contains nodes of type 0, so, to find such an inequality, we use Heuristic 2 to partition H into node sets H_i, each containing exactly one node of a nonzero type. There is, however, no guarantee that a violated, lifted 2-cover inequality is produced, even if one exists.

A sketch of our heuristic for separating lifted 2-cover constraints follows.

Heuristic 4. Separating Lifted 2-Cover Constraints

Given (G, r) and a point $y \in \mathbb{R}^E$.

Step 1. Shrink one-paths, as described in Section 4.7.

Step 2. Using the Padberg-Rao algorithm, find H, T with "low" values of (10).

Step 3. Try to transform each H and T found in Step 2 into a (hopefully violated) lifted 2-cover constraint, as described in Section 4.8.

In the remaining sections we will give some further details for Steps 1 and 3.

4.7. Shrinking One-Paths

We can shrink some edges according to criteria similar to those used for the separation of partition inequalities. Many such shrinking criteria are known for the TSP-case (such as alternating paths and one-paths, (see Padberg and Grötschel 1985, and Padberg and Rinaldi 1990) but not all apply to our two-connected case. For instance, the concept of alternating paths does not carry over because in our case the degree equations $x(\delta(v)) = 2$ are not necessarily satisfied. But we can shrink certain *one-paths*, i.e., paths P, where all edges $e \in P$ have $y_e = 1$ and all nodes $v \in P$ (except the endnodes) satisfy $y(\delta(v)) = 2$.

All one-paths between two nodes of type 2 may be shrunk to a single edge e having weight $y_e = 1$. Also, all one-paths where one endpoint is of type one or zero, and has at most two incident edges with nonzero weight (for instance, the leaf of a tree of one-edges), may be shrunk into a single node (see Figure 9).

4.8. Converting Odd Cuts into Lifted 2-Cover Constraints

Suppose that we have found $H \subseteq V$ and an odd subset T of $\delta(H)$ such that (10) has a "low" value. Our aim is to transform this into a (hopefully violated) lifted 2-cover constraint.

We choose H or $V \backslash H$ (usually we try both) as the handle of the lifted 2-cover inequality, and we use the given set T as its teeth. Using Heuristic 2 we find a partition of H into node sets H_i, $i = 1, \ldots, p$, each containing at least one node of a type at least 1.

We still have to adjust the lifted 2-cover inequality to meet the requirement that no more than one tooth is incident to a set H_i with $r(H_i) = 2$, and that no tooth is incident to a node set H_i with $r(H_i) = 1$. We do this in the following way:

- If exactly two teeth are incident to a node set H_i, we set $H := H \backslash H_i$ and $T := T \backslash \delta(H_i)$. If $|T| = 1$ we discard the current lifted 2-cover inequality.
- If more than two teeth are incident to some H_i, or if some tooth is incident to H_i with $r(H_i) = 1$, we also discard the current lifted 2-cover inequality.

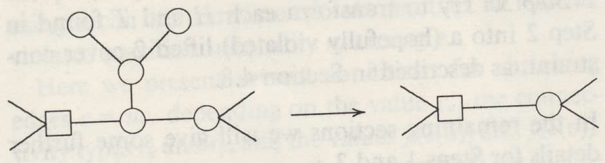

Figure 9. Shrinking one-paths.

We now give a short description of what we implemented of the separation routines listed above. First, we implemented the Gomory-Hu algorithm for separating the cut constraints. (This is an exact algorithm.) After we observed that many cut constraints were too weak (in the sense that they did not define facets), we implemented the test for Steiner bridges and, as an alternative, Heuristic 2. To reduce the graph sizes to which the Gomory-Hu algorithm is applied, we implemented the various shrinking criteria outlined above. Some of the routines used for the separation of partition inequalities, like the Gomory-Hu algorithm, Heuristic 2, and some of the shrinking procedures, were also useful to separate node partition and lifted 2-cover constraints. So we actually put all these ideas to work in our implementation.

5. COMPUTATIONAL RESULTS

The aim of our work was to develop a cutting plane algorithm that solves problems of the type and size that come up in the design of survivable telephone networks in fiber optic technology. We knew beforehand that the number of hubs (nodes) considered in practical applications is relatively small (at most 200) and that the networks (graphs) of possible direct fiber links (edges) are quite sparse. Thus, we had good hopes that the preprocessing and LP-relaxation techniques would provide very good lower bounds for the true optima in short computation time. We now report how well our cutting plane algorithm performed on these problems.

To test our code, network designers at Bell Communications Research provided the data (nodes, possible direct links, costs for establishing a link) of 7 real networks that were considered typical for this type of application. The sizes ranged from 36 nodes and 65 edges to 116 nodes and 173 edges; see Table I. The problem instances **LATADL, LATADS,** and **LATADSF** are defined on the same graph G. The edges have the same costs in each case, but the node types vary. Moreover, in **LATADSF**, 40 edges were required to be in the solution. (The purpose of these problem variations was to see how the cost would change under different scenarios. This approach is typically used in practice, where several alternative solutions are usually investigated before the planner selects a final solution.)

Table I provides information about the problems. Column 1 contains the problem names. For the original graphs, columns 2, 3, and 4 contain the number of nodes of type 0, 1, and 2, respectively; column 5 lists the total number of nodes, column 6 the number

Table I
Problem Descriptions for the Original and Reduced Graphs

	Original Graph					Reduced Graph				
Problem	0	1	2	Nodes	Edges	0	1	2	Nodes	Edges
LATADMA	0	12	24	36	65/0	0	6	15	21	46/4
LATA1	8	65	14	77	112/0	0	10	14	24	48/2
LATA5S	0	31	8	39	71/0	0	15	8	23	50/0
LATA5L	0	36	10	46	98/0	0	20	9	29	77/1
LATADSF	0	108	8	116	173/40	0	28	11	39	86/26
LADADS	0	108	8	116	173/0	0	28	11	39	86/3
LATADL	0	84	32	116	173/0	0	11	28	39	86/6

of edges and the number of edges required to be in any solution (the forced edges). All graphs were analyzed by our preprocessing procedures described in Section 2. Preprocessing was very successful. In fact, in every case, the decomposition and fixing techniques ended up with a single, much smaller graph obtained from the original graph by splitting off side branches consisting of nodes of type 1, replacing paths where all interior nodes are of type 2 by forced edges, etc. The data of the resulting reduced graphs are listed in columns 6, . . . , 10 of the table.

To give a visual impression of the problem topologies and the reductions achieved Figure 10 shows a picture of the original graph of the **LATADL** problem (with 32 nodes of type 2 and 84 nodes of type 1), and Figure 11 is a picture of the reduced graph (with 39 nodes and 86 edges) after preprocessing. The nodes of type 2 are displayed by squares, and the nodes of type 1 are displayed by circles. The six forced edges that have to be in any feasible solution are drawn in bold.

LATA1 is a **2ECON** problem, while the other six instances are **2NCON** problems. All optimum solutions of the **2ECON** versions turned out to satisfy all node-survivability constraints and thus were optimum solutions of the original **2NCON** problems—with one exception. In **LATA5L**, one node is especially attractive because many edges with low cost lead to it. This node is an articulation node of the optimum **2ECON** solution. In the following, **LATA5LE** is the **2ECON** version of **LATA5L**.

We now provide some details of our algorithm and its implementation. In a preprocessing phase we try to decompose and reduce the given problems using the methods described in Section 2. The result of this procedure is a graph (or a list of graphs) that is not decomposable. The cutting plane algorithm is called for each such graph.

Our cutting plane algorithm follows the standard approach (see the Cutting Plane Algorithm). We use the framework of a (general) branch-and-cut algorithm that is currently being developed by Michael Jünger. The LP-solver used is a research version of the CPLEX-code provided to us by Bixby (1991). This is a very fast implementation of the simplex algorithm.

Since the number of variables of our test problems is relatively small, we do not employ any techniques (other than preprocessing) to eliminate variables from a current LP.

In 5 of the 8 test problems, the cutting plane algorithm produced an optimum solution. The other three cases were solved by branch and cut. This consists of choosing a branching variable, setting some variables to their upper and lower bounds according to their reduced costs and some logical implications, and finding new cutting planes for the modified problem. The branch-and-cut tree is traversed in depth-first search fashion. We did not run heuristics to provide a good initial upper bound for the enumeration phase or to turn fractional solutions into feasible integral solutions to improve the intermediate upper bounds. We simply wanted to test the general method (developed by Michael Jünger) for this particular case. It did, as Table II shows, quite well. Clearly, considerable improvements can be achieved by implementing problem-specific modifications and adding various heuristics. We do not go into further details of the branch-and-cut procedure because this is very technical (and even a rough outline would require a lot of space) and the emphasis is on the cutting plane algorithm in this paper.

Our code also contains an option to add further cuts manually. For instance, we were able to solve **LATA5L** by manually adding partition, node partition, and 2-cover inequalities that could not be found automatically; for **LATADL** we had to use, among others, a certain inequality to achieve the optimal solution without branching, and the best lower bound for **LATADS** that we could reach by adding any valid inequalities known to us is still one unit away from

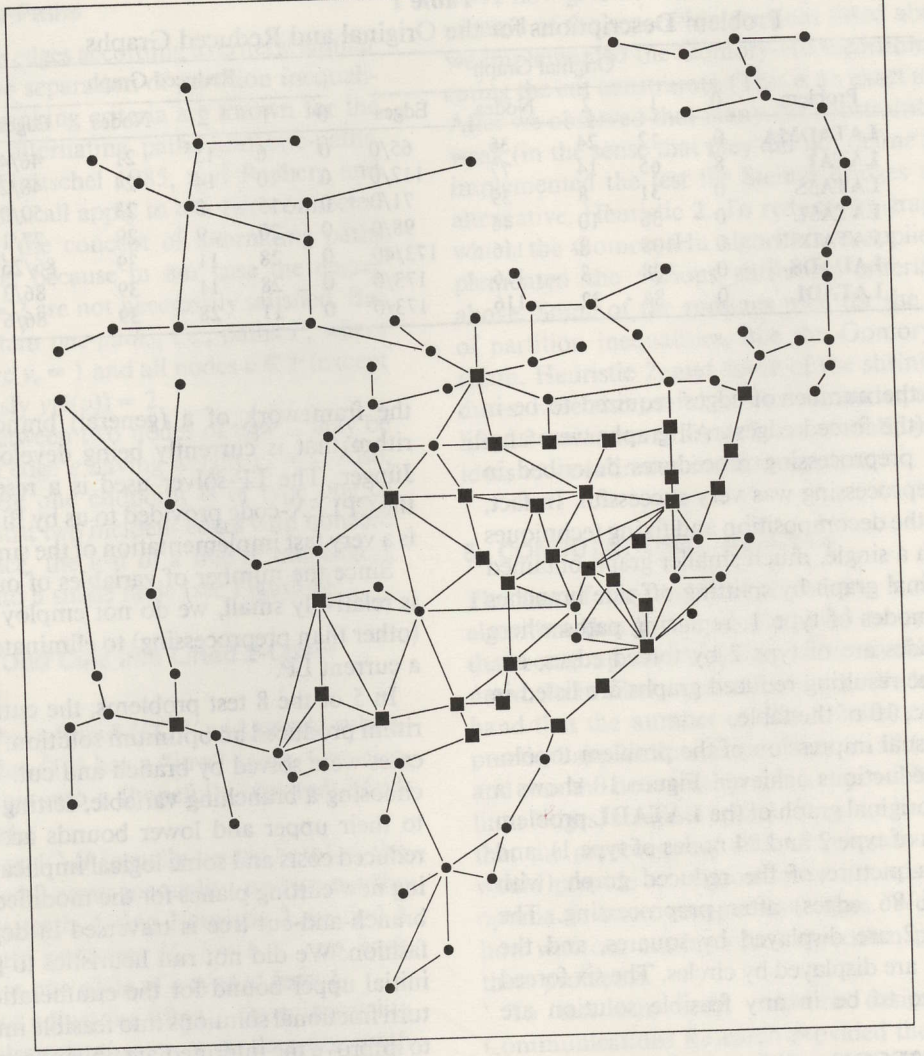

Figure 10. Original graph of the **LATADL** problem.

the optimal value. So our polyhedral description of 2ECON(G; r) and 2NCON(G; r) is good enough to achieve very good lower bounds, but not sufficient to always find the optimal solution without resorting to an enumeration phase like branch and cut.

Table II contains some data about the performance of our code on the 8 test instances. The entries from left to right are:

IT	the number of iterations (= calls of the LP solver);
P	the number of partition inequalities (3) added to the initial LP (7);
NP	the number of node partition inequalities (4) added to the initial LP (7);
2C	the number of lifted 2-cover inequalities (5) added to the initial LP (7);
C	the value of the optimum solution after termination of the cutting plane phase;
COPT	the optimum value;
GAP	100 × (COPT − C)/COPT (= the percent relative error at the end of the cutting plane phase);
T	the total running time including input, output, preprocessing, etc. of the cutting plane phase (not including branch and cut), in rounded seconds on a SUN 3/60 workstation (a 3-MIPS machine);
BN	the number of branch-and-cut nodes generated;

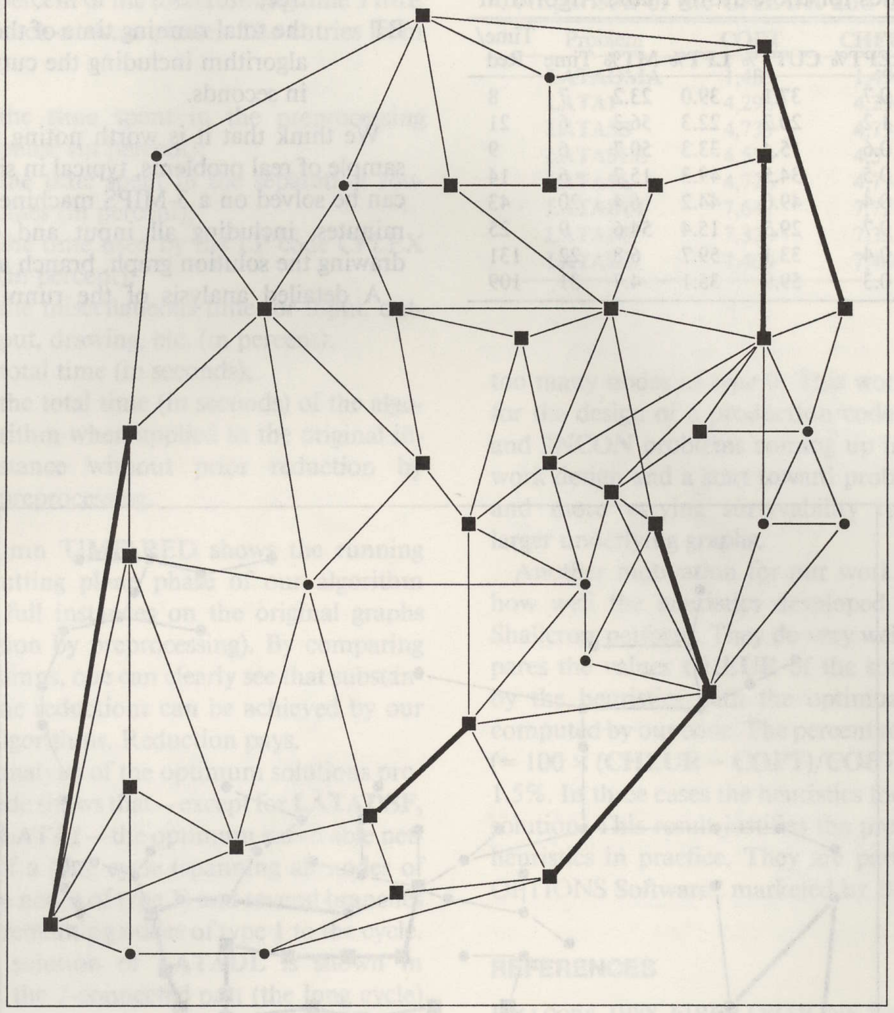

Figure 11. Reduced graph of the **LATADL** problem.

Table II
Data for Eight Test Problems

Problem	IT	P	NP	2C	C	COPT	GAP	T	BN	BD	BT
LATADMA	11	49	3	5	1,489.00	1,489	0.00	7			
LATA1	4	43	0	1	4,296.00	4,296	0.00	6			
LATA5S	4	53	0	0	4,739.00	4,739	0.00	6	—	—	—
LATA5LE	6	67	0	0	4,574.00	4,574	0.00	6			
LATA5L	16	128	8	1	4,679.00	4,726	0.99	20	19	7	80
LATADSF	6	26	0	0	7,647.00	7,647	0.00	9	—	—	—
LATADS	9	187	0	2	7,300.00	7,320	0.27	22	20	6	112
LATADL	14	160	0	28	7,378.25	7,400	0.33	31	6	3	69

Table III
Running Times for the Cutting Plane Algorithm

Problem	PREPT%	CUTT%	LPT%	MT%	Time	Time/ Red
LATADMA	0.7	37.1	39.0	23.2	7	8
LATA1	1.2	20.2	22.3	56.3	6	21
LATA5S	0.6	15.4	33.3	50.7	6	9
LATA5LE	0.5	34.5	49.3	15.7	6	14
LATA5L	0.4	49.0	44.2	6.4	20	43
LATADSF	0.7	29.3	15.4	54.6	9	25
LATADS	0.4	33.1	59.7	6.8	22	131
LATADL	0.3	59.9	35.1	4.7	31	109

BD	the maximum depth of the branch-and-cut tree;
BT	the total running time of the branch-and-cut algorithm including the cutting plane phase, in seconds.

We think that it is worth noting that each of this sample of real problems, typical in size and structure, can be solved on a 3-MIPS machine in less than two minutes, including all input and output routines, drawing the solution graph, branch and cut, etc.

A detailed analysis of the running times of the

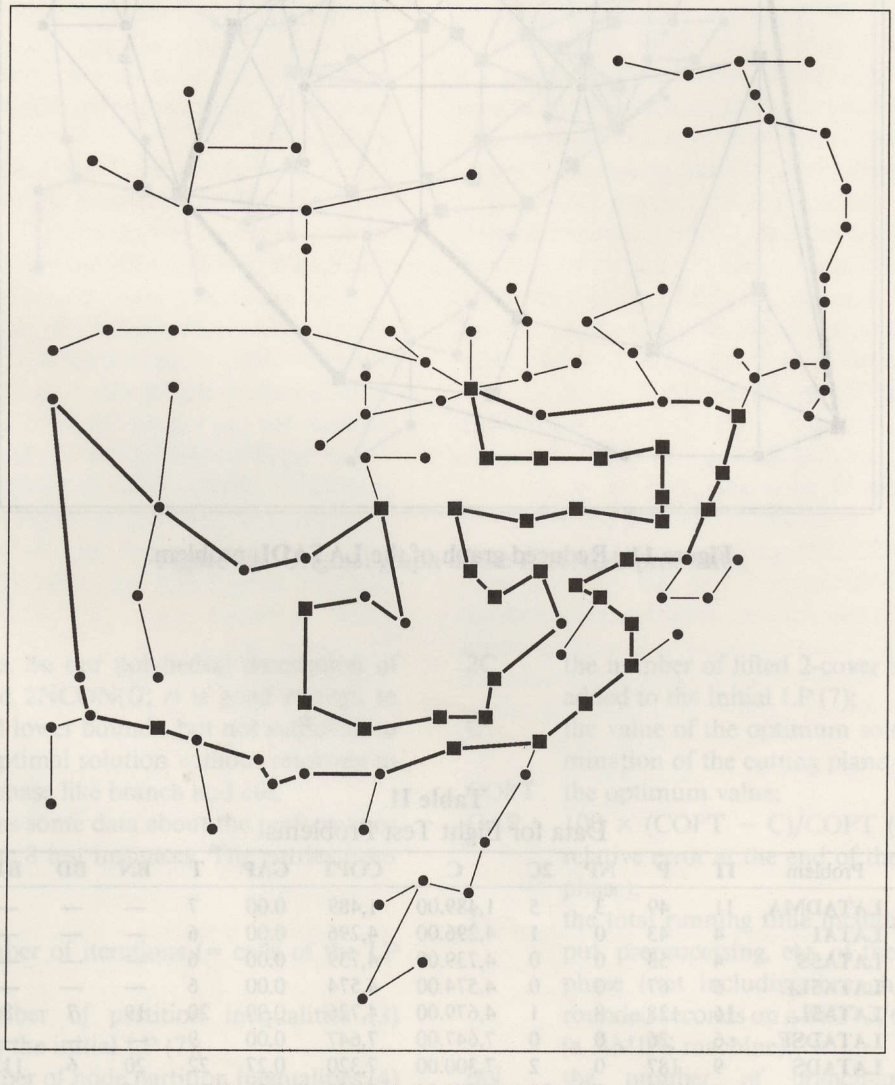
Figure 12. Solution of the LATADL problem.

cutting plane phase is given in Table III. All times reported are in percent of the total running time TIME (without the branch-and-cut phase). The entries from left to right are:

PREPT	the time spent in the preprocessing phase (in percent);
CUTT	the time spent in the separation routines (in percent);
LPT	the time used by the LP code CPLEX (in percent);
MT	the miscellaneous time for input, output, drawing, etc. (in percent);
TIME	total time (in seconds);
TIME\RED	the total time (in seconds) of the algorithm when applied to the original instance without prior reduction by preprocessing.

The last column TIME\RED shows the running times of the cutting plane phase of our algorithm applied to the full instances on the original graphs (without reduction by preprocessing). By comparing the last two columns, one can clearly see that substantial running time reductions can be achieved by our preprocessing algorithms. Reduction pays.

A structural analysis of the optimum solutions produced by our code shows that—except for **LATADSF**, **LATA5L**, and **LATA1**—the optimum survivable networks consist of a long cycle (spanning all nodes of type 2 and some nodes of type 1) and several branches connecting the remaining nodes of type 1 to the cycle. The optimum solution of **LATADL** is shown in Figure 12, with the 2-connected part (the long cycle) drawn in bold.

We ran a few tests on randomly generated problems of higher density. Here our code performed reasonably but not as well. (That is not of great importance because our goal was to solve real-world and not random problems.) More serious is a dramatic increase in running time when many nodes of type 0 are added. In this case, it takes very long before the intermediate fractional solutions become connected. We think that for such cases new separation heuristics have to be developed that perform a more sophisticated structural analysis of the given instance. But the problems that we address here and that come up in the design of fiber optic telephone networks have very few, if any, nodes of type 0.

Our computational experiments show that our approach produces very good lower bounds and even optimum solutions in the initial cutting plane phase for problem instances that are sparse and do not have

Table IV
Comparison of the **CHEUR** and **COPT** Solutions

Problem	COPT	CHEUR	GAP
LATADMA	1,489	1,494	0.34
LATA1	4,296	4,296	0.00
LATA5S	4,739	4,739	0.00
LATA5LE	4,574	4,574	0.00
LATA5L	4,726	4,794	1.44
LATADSF	7,647	7,727	1.05
LATADS	7,320	7,361	0.56
LATADL	7,400	7,460	0.81

too many nodes of type 0. This work is a good basis for the design of a production code for the **2ECON** and **2NCON** problems coming up in fiber optic network design and a start toward problems with higher and more varying survivability requirements and larger underlying graphs.

Another motivation for our work was to find out how well the heuristics developed in Monma and Shallcross perform. They do very well. Table IV compares the values **CHEUR** of the solutions produced by the heuristics with the optimum values **COPT** computed by our code. The percent relative error **GAP** (= 100 × (**CHEUR** − **COPT**)/**COPT**) is always below 1.5%. In three cases the heuristics found an optimum solution. This result justifies the present use of these heuristics in practice. They are part of the "FIBER OPTIONS Software" marketed by Bellcore (1988).

REFERENCES

Bellcore, 1988. FIBER OPTIONS, Software for Designing Survivable Optimal Fiber Networks. Software Package, Bellcore, Morristown, N.J.

Bixby, R. E. 1991. Implementing the Simplex Method: The Initial Basis. Technical Report TR90-32, Department of Mathematical Sciences, Rice University, Houston, Texas.

Cardwell, R. H., C. L. Monma and T. H. Wu. 1989. Computer-Aided Design Procedures for Survivable Fiber Optic Networks. *IEEE Select. Areas Commun.* **7**, 1188–1197.

Cardwell, R. H., H. Fowler, H. L. Lemberg and C. L. Monma. 1988. Determining the Impact of Fiber Optic Technology on Telephone Network Design. *Bellcore Exchange Magazine,* March/April, 27–32.

Cook, W., and W. R. Pulleyblank. 1987. Linear Systems for Constrained Matching Problems. *Math. Opns. Res.* **12**, 97–120.

Cornuéjols, G., J. Fonlupt and D. Naddef. 1988. The Traveling Salesman Problem on a Graph and

Some Related Integer Polyhedra. *Math. Prog.* **33,** 1–27.

DAHLHAUS, E., D. S. JOHNSON, C. H. PAPADIMITRIOU, P. SEYMOUR AND M. YANNAKAKIS. 1983. The Complexity of Multiway Cuts. Unpublished Extended Abstract.

GOLDBERG, A. V., AND R. E. TARJAN. 1987. A New Approach to the Maximum Flow Problem. In *Proceedings of the Eighth Annual ACM Symposium on the Theory of Computing.*

GOMORY, R. E., AND T. C. HU. 1961. Multi-Terminal Network Flows. *SIAM J. Appl. Math.* **9,** 551–570.

GRÖTSCHEL, M., AND C. L. MONMA. 1990. Integer Polyhedra Associated With Certain Network Design Problems With Connectivity Constraints. *SIAM J. Discr. Math.* **3,** 502–523.

GRÖTSCHEL, M., AND M. W. PADBERG. 1985. Polyhedral Theory. In *The Traveling Salesman Problem,* E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. Shmoys (eds.). John Wiley, Chichester, 251–305.

GRÖTSCHEL, M., C. L. MONMA AND M. STOER. 1989. Facets for Polyhedra Arising in the Design of Communication Networks With Low-Connectivity Constraints. *SIAM J. Opt.* (to appear).

GRÖTSCHEL, M., C. L. MONMA AND M. STOER. 1991. Polyhedral Approaches to Network Survivability. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **5,** AMS, 121–141.

GRÖTSCHEL, M., L. LOVÁSZ, AND A. SCHRIJVER. 1988. *Geometric Algorithms and Combinatorial Optimization.* Springer, Berlin.

GUSFIELD, D. 1987. Very Simple Algorithms and Programs for all Pairs Network Flow Analysis. Computer Science Division, University of California, Davis, Calif.

HOPCROFT, J., AND R. E. TARJAN. 1973a. Algorithm 447: Efficient Algorithms for Graph Manipulation. *Comm. ACM* **16,** 372–378.

HOPCROFT, J., AND R. E. TARJAN. 1973b. Dividing a Graph into Triconnected Components. *SIAM J. Comput.* **2,** 135–158.

MAHJOUB, A. R. 1988. Two Edge Connected Spanning Subgraphs and Polyhedra. Report No 88520-OR, Institut für Operations Research, Universität Bonn, Germany.

MONMA, C. L., AND D. F. SHALLCROSS. 1989. Methods for Designing Communication Networks With Certain Two-Connected Survivability Constraints. *Opns. Res.* **37,** 531–541.

MONMA, C. L., B. S. MUNSON AND W. R. PULLEYBLANK. 1990. Minimum-Weight Two-Connected Spanning Networks. *Math. Prog.* **46,** 153–171.

PADBERG, M. W., AND M. GRÖTSCHEL. 1985. Polyhedral Computations. In *The Traveling Salesman Problem,* E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. Shmoys (eds.). John Wiley, Chichester, 307–360.

PADBERG, M. W., AND M. R. RAO. 1982. Odd Minimum Cut-Sets and *b*-Matchings. *Math. Opns. Res.* **7,** 67–80.

PADBERG, M. W., AND G. RINALDI. 1990. Facet Identification for the Symmetric Traveling Salesman Problem. *Math. Prog.* **47,** 219–257.

PRODON, A., T. M. LIEBLING AND H. GRÖFLIN. 1985. Steiner's Problem on Two-Trees. Report No. 850315, Department of Mathematics, École Polytechnique Féderale de Lausanne, Switzerland.