Capacitated network design using general flow-cutset inequalities *

Christian Raack† Arie M.C.A. Koster‡ Sebastian Orlowski†
Roland Wessäly†

Abstract

This paper deals with directed, bidirected, and undirected capacitated network design problems. Using mixed integer rounding (MIR), we generalize flow-cutset inequalities to these three link types and to an arbitrary modular link capacity structure, and propose a generic separation algorithm. In an extensive computational study on 54 instances from the Survivable Network Design Library (SNDlib), we show that the performance of CPLEX can significantly be enhanced by this class of cutting planes. The computations reveal the particular importance of the subclass of cutset inequalities.

Keywords: general flow-cutset inequalities, capacitated network design, mixed integer rounding, SNDlib

MSC: 90C11, 90C35, 90C57, 90B18

1 Introduction

The task of capacitated network design is to assign capacity to the links of a potential network topology by selecting capacity modules (wavelength channels, leased lines, STM-N capacities) from a discrete set such that given communication demands can be satisfied and total installation cost is minimized.

In this paper we revisit polyhedral approaches based on cutsets for directed, bidirected and undirected link models [2, 6, 8, 13, 14, 15]. Flow-cutset inequalities that have been studied by Atamtürk [2] for the directed case are generalized to bidirected and undirected link models in this paper, and their efficiency within a branch-and-cut framework is investigated. A generic separation algorithm, that unifies known separation heuristics [2, 5, 6, 7, 9, 15] for all three link models and an arbitrary modular link capacity structure, is presented.

Our computational study comprises 54 instances covering all SNDlib [18] networks, the link types DIRECTED, UNDIRECTED, and BIDIRECTED, as well as the capacity models MODULAR and EXPLICIT defined in SNDlib. In all 54 cases, separating general flow-cutset inequalities substantially improves the performance of CPLEX. In particular, 27 instances can be solved to optimality within one hour of computation time, compared to 17 with the default settings of CPLEX. The best solutions are available at the SNDlib homepage `http://sndlib.zib.de`. It is the first time that the practical strength of these cutting planes for capacitated network design problems is proven using such a variety of realistic instances. These results confirm their theoretical importance, shown in an accompanying polyhedral study [19].

A network instance is given by a directed graph $G = (V, A)$ (DIRECTED link model) or an undirected graph $H = (V, E)$ (BIDIRECTED and UNDIRECTED link model), a set M of installable modules, and a set K of commodities. For undirected graphs we define $G = (V, A)$ to be the digraph that

*This research has been supported by the German Ministry of Education and Research (BMBF) within the EIBONE project under contract number 01BP567.

†Zuse Institute Berlin (ZIB), Takustr. 7, D-14195 Berlin, {raack,orlowski,wessaely}@zib.de

‡University of Warwick, Centre for Discrete Mathematics and its Applications (DIMAP), Coventry CV4 7AL, United Kingdom, Arie.Koster@wbs.ac.uk

is obtained by introducing a pair of antiparallel arcs e^+ and e^- for every edge e in E. We set $A := \{e^+ = (i,j),\, e^- = (j,i) : e = \{i,j\} \in E\}$. For all models, $\delta_G^+(v)$ and $\delta_G^-(v)$ denote the arcs in A that have v as their source and target node, respectively. A module $m \in M$ has a capacity $c^m \in \mathbb{Z}_+ \backslash \{0\}$. Let x_a^m, x_e^m be the number of installed modules of type $m \in M$ on arc $a \in A$ or edge $e \in E$, respectively.

With every commodity $k \in K$ we associate a vector $d^k \in \mathbb{Z}^V$ of demands such that $\sum_{v \in V} d_v^k = 0$, assuming a fractional multi-commodity flow routing. There are mainly two approaches related to the definition of commodities and demands. The first is to consider an individual commodity for every point-to-point demand, resulting in $|K| \in O(|V|^2)$ commodities. In our formulations, point-to-point traffic demands are assumed to be aggregated at their source nodes. Thus, every commodity $k \in K$ can be identified with a unique source node s such that $d_s^k > 0$ and $d_v^k \leq 0$ for all $v \in V \backslash \{s\}$. There might be several target nodes with negative demand value. This approach may lead to a significantly reduced problem size with $|K| \in O(|V|)$. The flow of a commodity can be split among several paths and it is allowed to be fractional. Let $f_a^k \in \mathbb{R}_+$ be the flow variable for arc $a \in A$ and commodity $k \in K$. We consider the flow conservation constraints

$$\sum_{a \in \delta_G^+(v)} f_a^k - \sum_{a \in \delta_G^-(v)} f_a^k = d_v^k \qquad \forall v \in V,\, k \in K \tag{1}$$

and, depending on the link type, one of the following sets of capacity constraints

$$\sum_{k \in K} f_a^k \leq \sum_{m \in M} c^m x_a^m \qquad \forall a \in A\,, \tag{2}$$

$$\max\Big\{ \sum_{k \in K} f_{e^+}^k, \sum_{k \in K} f_{e^-}^k \Big\} \leq \sum_{m \in M} c^m x_e^m \qquad \forall e \in E\,, \tag{3}$$

$$\sum_{k \in K} (f_{e^+}^k + f_{e^-}^k) \leq \sum_{m \in M} c^m x_e^m \qquad \forall e \in E\,. \tag{4}$$

The network design polyhedra for the link types Directed, Bidirected, and Undirected are given by

$$\begin{aligned}
P_{di} &= \operatorname{conv}\big\{(f,x) \in \mathbb{R}_+^{A \times K} \times \mathbb{Z}_+^{A \times M} : (f,x) \text{ satisfies (1), (2)}\big\}\,,\\
P_{bi} &= \operatorname{conv}\big\{(f,x) \in \mathbb{R}_+^{A \times K} \times \mathbb{Z}_+^{E \times M} : (f,x) \text{ satisfies (1), (3)}\big\}\,,\\
P_{un} &= \operatorname{conv}\big\{(f,x) \in \mathbb{R}_+^{A \times K} \times \mathbb{Z}_+^{E \times M} : (f,x) \text{ satisfies (1), (4)}\big\}\,.
\end{aligned}$$

Notice that the capacity constraints (3) for P_{bi} can be expressed by two linear inequalities for each link, that P_{bi} is a relaxation of P_{un}, and that the constraint matrices and right-hand side vectors are integral. In addition to this Modular capacity model, we also consider the Explicit capacity model where the constraints $\sum_{m \in M} x_a^m \leq 1$ for $a \in A$ are added to the Directed formulation and the Bidirected and Undirected formulations are extended by $\sum_{m \in M} x_e^m \leq 1$ for all $e \in E$. These generalized upper bound constraints ensure that at most one module is installed on every network link.

We study network design problems whose objective is to minimize a linear function incorporating flow and module cost over one of the defined polyhedra. We focus on these core problems of capacitated network design and do not consider any further problem-specific constraints such as survivability requirements or node capacities. Depending on the application, integral or single-path routing may be required; the polyhedra under consideration relax both cases at a uniform level. The general flow-cutset inequalities derived in this paper can easily be generalized to arc-dependent module sets and to models with preinstalled capacity.

This paper is organized as follows. In Section 2 we introduce the necessary notation and present general flow-cutset inequalities for the three link types. It is shown how to derive these cutting planes by mixed integer rounding (MIR) and how to strengthen so-called simple flow-cutset inequalities. Section 3 focuses on the separation problem. We present a generic separation procedure that can be used for all considered models and an arbitrary number of modules. In Section 4 we report on the

effect of the generated inequalities in reducing computation times and gaps for a large set of test instances taken from SNDlib. Section 5 contains some concluding remarks. A literature review can be found at the end of Section 2 and at the beginning of Section 3.

2 General flow-cutset inequalities

In this section, we generalize flow-cutset inequalities that have been presented in [2, 6, 8, 13, 14, 15] for special cases of the network design problems considered in this paper. We derive them by aggregating model constraints on a cut in the network and applying a subadditive MIR function to the coefficients of the resulting inequality. As shown in [2, 19], these inequalities are facet-defining for the network design polyhedra P_{di}, P_{un}, and P_{bi} under rather general conditions. We will first introduce the subadditive MIR function that we use to derive strong inequalities. We will then develop the cut-based inequalities to which the function is applied. Special cases studied in the literature are reviewed at the end of this section.

Mixed integer rounding. Let $a, c, d \in \mathbb{R}$ with $c > 0$ and define $a^+ := \max(0, a)$ and $a^- := \min(0, a)$. Furthermore, let

$$r_{a,c} := a - c(\lceil \tfrac{a}{c} \rceil - 1) > 0 \tag{5}$$

be the remainder of the division of a by c if $\frac{a}{c} \notin \mathbb{Z}$, and c otherwise. A function $F : \mathbb{R} \to \mathbb{R}$ is called *subadditive* if $F(a) + F(b) \geq F(a + b)$ and *superadditive* if $F(a) + F(b) \leq F(a + b)$ for all $a, b \in \mathbb{R}$. We consider the function $F_{d,c} : \mathbb{R} \to \mathbb{R}$ defined by

$$F_{d,c}(a) := \lceil \tfrac{a}{c} \rceil r_{d,c} - (r_{d,c} - r_{a,c})^+.$$

Lemma 2.1. *The function $F_{d,c}$ is subadditive and nondecreasing with $F_{d,c}(0) = 0$. If $\frac{d}{c} \notin \mathbb{Z}$, then $\bar{F}_{d,c}(a) := \lim_{t \searrow 0} \frac{F_{d,c}(at)}{t} = a^+$ for all $a \in \mathbb{R}$. If otherwise $\frac{d}{c} \in \mathbb{Z}$, then $\bar{F}_{d,c}(a) = F_{d,c}(a) = a$ for all $a \in \mathbb{R}$.*

Proof. Define $f_x := x - \lfloor x \rfloor$ for $x \in \mathbb{R}$. For $b \in \mathbb{R}$, let $\varphi_b : \mathbb{R} \to \mathbb{R}$, defined by

$$\varphi_b(a) := \lfloor a \rfloor + \frac{(f_a - f_b)^+}{1 - f_b}$$

be the MIR-function for $\leq$-base-inequalities with right-hand side b. According to Nemhauser and Wolsey [17, §II.1.7], the function φ_b is superadditive and nondecreasing with $\varphi_b(0) = 0$ and if $b \notin \mathbb{Z}$ then

$$\lim_{t \searrow 0} \frac{\varphi_b(at)}{t} = \frac{a^-}{1 - f_b}$$

for all $a \in \mathbb{R}$.

Using the relation $cf_{\frac{-a}{c}} = c - r_{a,c}$ which follows by definition (5) of the operator $r_{a,c}$, it holds that

$$\begin{aligned} -r_{d,c} \cdot \varphi_{-\frac{d}{c}}(-\tfrac{a}{c}) = -r_{d,c}\lfloor -\tfrac{a}{c} \rfloor - r_{d,c}\frac{c\big(f_{-\frac{a}{c}} - f_{-\frac{d}{c}}\big)^+}{c - cf_{-\frac{d}{c}}} &= r_{d,c}\lceil \tfrac{a}{c} \rceil - r_{d,c}\frac{(r_{d,c} - r_{a,c})^+}{r_{d,c}} \\ &= F_{d,c}(a). \end{aligned}$$

It follows that $F_{d,c}(a)$ is subadditive and nondecreasing with $F_{d,c}(0) = 0$. If $\frac{d}{c} \notin \mathbb{Z}$, then

$$\lim_{t \searrow 0} \frac{F_{d,c}(at)}{t} = -r_{d,c} \lim_{t \searrow 0} \frac{\varphi_{-\frac{d}{c}}(-\frac{a}{c}t)}{t} = -r_{d,c}\frac{c(\frac{-a}{c})^-}{c - cf_{-\frac{d}{c}}} = -(-a)^- = a^+.$$

If $\frac{d}{c} \in \mathbb{Z}$, then $r_{d,c} = c$ and $F_{d,c}(a) = \lceil \frac{a}{c} \rceil c - c + r_{a,c} = a = \bar{F}_{d,c}(a)$ for all $a \in \mathbb{R}$. □

Lemma 2.2. *$|F_{d,c}(a)| \le |a|$ for all $a \in \mathbb{R}$. Moreover, $a, c, d \in \mathbb{Z}$ implies $F_{d,c}(a), \bar{F}_{d,c}(a) \in \mathbb{Z}$.*

Proof. If a, c, $d \in \mathbb{Z}$, then $F_{d,c}(a)$, $\bar{F}_{d,c}(a) \in \mathbb{Z}$ by definition. It remains to show that $|F_{d,c}(a)| \le |a|$ for any $a \in \mathbb{R}$. First assume $a > 0$. If $r_{d,c} \le r_{a,c}$, then

$$0 \le F_{d,c}(a) = r_{d,c}\lceil \tfrac{a}{c} \rceil \le r_{a,c}\lceil \tfrac{a}{c} \rceil = a - (\lceil \tfrac{a}{c} \rceil - 1)(c - r_{a,c}) \le a$$

since $\lceil \frac{a}{c} \rceil \ge 1$ and $0 \le r_{a,c} \le c$. Now let $r_{d,c} > r_{a,c}$. Then $r_{a,c} < c$ and

$$0 \le F_{d,c}(a) = r_{d,c}\lceil \tfrac{a}{c} \rceil - (r_{d,c} - r_{a,c}) = r_{d,c}\lfloor \tfrac{a}{c} \rfloor + r_{a,c} \le c\lfloor \tfrac{a}{c} \rfloor + r_{a,c} = a.$$

For $a < 0$, the result follows from $F_{d,c}(a) = F_{-d,c}(-a) + a$. □

The function $F_{d,c}$ can be seen as the $1/c$-MIR function for $\ge$-base-inequalities with right-hand side d, scaled by the factor $r_{d,c}$. Similar subadditive and superadditive functions based on MIR have been considered for instance by Atamtürk [2, 3, 4] and Louveaux and Wolsey [12].

Lemma 2.3 ([17, Chapter II.1, Theorem 7.4])**.** *Let*

$$X := \Big\{(f, x) \in \mathbb{R}_+^{N_1} \times \mathbb{Z}_+^{N_2} : \sum_{j \in N_1} \gamma_j f_j + \sum_{j \in N_2} c_j x_j \ge d\Big\}$$

with N_1, N_2 being two finite index sets and γ_j, c_j, d rational numbers. If the function $F : \mathbb{R} \to \mathbb{R}$ is nondecreasing and subadditive on $\mathbb{R}$ with $F(0) = 0$ and $\bar{F}$ exists for all $j \in N_1$, then

$$\sum_{j \in N_1} \bar{F}(\gamma_j) f_j + \sum_{j \in N_2} F(c_j) x_j \;\ge\; F(d)$$

is valid for X.

General flow-cutset inequalities. By Lemma 2.1 and Lemma 2.3, applying the function $F_{d,c}$ to the coefficients of valid base inequalities yields new valid inequalities. In the sequel we will consider base inequalities for network design polyhedra that correspond to cuts of the underlying network. We first introduce the necessary notation.

Consider a cut defined by a subset S of the supply nodes V and let Q be a subset of the commodities K. For UNDIRECTED and BIDIRECTED link types, consider the undirected cut

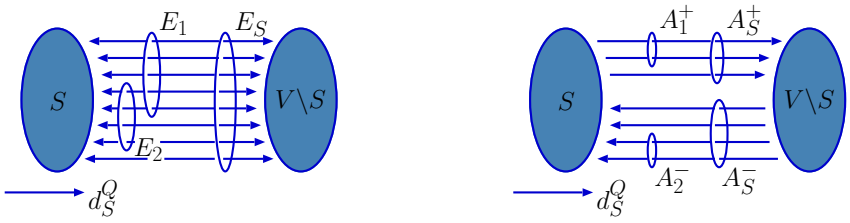
$$E_S := \{e \in E : i \in S,\, j \in V \backslash S\}$$

with subsets $E_1, E_2 \subseteq E_S$ (see Figure 1(a)). Similarly, for DIRECTED models, define

$$A_S^+ := \{(i,j) \in A : i \in S,\, j \in V \backslash S\} \quad \text{and} \quad A_S^- := \{(i,j) \in A : j \in S,\, i \in V \backslash S\},$$

and consider subsets $A_1^+ \subseteq A_S^+$, $A_2^- \subseteq A_S^-$, and $\bar{A}_1^+ := A_S^+ \backslash A_1^+$ (see Figure 1(b)). The sets A_1^+, $\bar{A}_1^+$, and A_2^- are also used for UNDIRECTED and BIDIRECTED models. In this case, the sets A_1^+, $\bar{A}_1^+$ correspond to forward flow on edges of E_1 and its complement with respect to the cut, whereas A_2^- corresponds to backward flow on edges of E_2.

Let $f^Q(A^*)$ denote the total flow on some subset A^* of the arcs A with respect to Q, i. e., $f^Q(A^*) := \sum_{k \in Q} \sum_{a \in A^*} f_a^k$, and let $x^m(A^*) := \sum_{a \in A^*} x_a^m$ be the total number of modules of type $m \in M$ on arcs of A^*. The value $x^m(E^*) := \sum_{e \in E^*} x_e^m$ is defined analogously for undirected edges $E^* \subseteq E$. In the following, the node-set S and the commodity subset Q are fixed. Let $d_S^Q := \sum_{v \in S} \sum_{k \in Q} d_v^k$ be the total demand value with respect to Q over the cut defined by S, where $d_S^k := d_S^{\{k\}}$ for $k \in K$. By switching between S and $V \backslash S$ we may assume that $d_S^Q \ge 0$ since $d_S^Q = -d_{V \backslash S}^Q$. Let $K_S^+ := \big\{k \in K : d_S^k > 0\big\}$ and $K_S^- := \big\{k \in K : d_S^k < 0\big\}$ be the commodity subsets with positive and negative demand over the cut, respectively.



(a) Undirected cut and edge-sets

(b) Directed cut and arc-sets

Figure 1: Network cuts

We will now develop a valid base inequality that is defined for S and Q. By aggregating all flow conservation constraints corresponding to nodes in S and commodities in Q we obtain

$$f^Q(A_S^+) - f^Q(A_S^-) \;\geq\; d_S^Q$$

Adding this aggregated flow conservation constraint to the aggregated DIRECTED capacity constraint

$$\sum_{m\in M} c^m x^m(A_1^+) \;\geq\; f^Q(A_1^+)$$

and to the non-negativity constraints for $A_S\backslash A_2^-$ results in

$$f^Q(\bar{A}_1^+) - f^Q(A_2^-) + \sum_{m\in M} c^m x^m(A_1^+) \;\geq\; d_S^Q.$$

By defining $\bar{f}^Q(A_2^-) := \sum_{m\in M} c^m x^m(A_2^-) - f^Q(A_2^-)$ to be the slack of the aggregated DIRECTED capacity constraint for A_2^-, we may write

$$f^Q(\bar{A}_1^+) + \bar{f}^Q(A_2^-) + \sum_{m\in M} c^m\big(x^m(A_1^+) - x^m(A_2^-)\big) \;\geq\; d_S^Q, \tag{6}$$

where $\bar{f}^Q(A_2^-)$ can be treated as a single nonnegative continuous variable. By construction, inequality (6) is valid for P_{di}. Notice that it is also valid for the linear relaxation of P_{di}. Considering edge-sets E_1 and E_2 gives a similar inequality for BIDIRECTED and UNDIRECTED problems:

$$f^Q(\bar{A}_1^+) + \bar{f}^Q(A_2^-) + \sum_{m\in M} c^m\big(x^m(E_1) - x^m(E_2)\big) \;\geq\; d_S^Q. \tag{7}$$

Notice that in contrast to the arc-sets A_1^+ and A_2^-, the edge-sets E_1 and E_2 are not necessarily disjoint. Applying the MIR function $F_{d,c}$ with $d = d_S^Q$ and $c = c^t$ for $t \in M$ to the coefficients of (6) and (7) leads to the class of *general flow-cutset inequalities*.

Proposition 2.4. *For any $t \in M$, let $F_t := F_{d_S^Q,c^t}$ and $\bar{F}_t := \bar{F}_{d_S^Q,c^t}$. The general flow-cutset inequality*

$$f^Q(\bar{A}_1^+) - f^Q(A_2^-) + \sum_{m\in M} F_t(c^m)x^m(A_1^+) + \sum_{m\in M} (c^m + F_t(-c^m))x^m(A_2^-) \;\geq\; F_t(d_S^Q) \tag{8}$$

is valid for P_{di}, whereas the following general flow-cutset inequality is valid for P_{bi} and P_{un}:

$$f^Q(\bar{A}_1^+) - f^Q(A_2^-) + \sum_{m\in M} F_t(c^m)x^m(E_1) + \sum_{m\in M} (c^m + F_t(-c^m))x^m(E_2) \;\geq\; F_t(d_S^Q). \tag{9}$$

Proof. Applying F_t to all coefficients of module variables and $\bar{F}_t$ to all coefficients of flow variables in (6) and (7) yields inequalities (8) and (9), respectively. Notice that $\bar{F}_t(1) = 1$. The proposition follows then by Lemma 2.1, Lemma 2.3 and resubstituting $\bar{f}^Q(A_2^-)$. □

In the light of [16], general flow-cutset inequalities are derived by *aggregating*, *substituting*, *scaling* and MIR. In fact, the function F_t is responsible for the last two steps. It scales the base inequalities (6), (7) with $1/c^t$ and then applies MIR. The resulting inequality is additionally rescaled to obtain integer coefficients. The MIR procedure to obtain flow-cutset inequalities is similar to the one presented by Louveaux and Wolsey [12] for single node flow sets and flow-cover inequalities. The only difference is that we do not complement capacity variables because they are not bounded. There are also relations to the concept of simultaneous (subadditive) lifting [2, 3, 4, 12]. In particular, the function F_t is identical to the lifting function ϕ_t^+ used in [2] to lift flow-cutset inequalities from the single-module to the multi-module case.

If d_S^Q is an integer multiple of c^t, then Lemma 2.1 implies that $F_t(a) = a$ for all $a \in \mathbb{R}$ and the inequalities (8) and (9) are identical to the trivially valid base inequalities (6) and (7). By Lemma 2.2, $F_{d,c}(a), \bar{F}_{d,c}(a)$ are integral if a, c, and d are integral. Moreover, $|F_{d,c}(a)| \leq |a|$ holds for all $a \in \mathbb{R}$. This means that the considered inequalities have small integral coefficients as long as capacities and demands are small and integral. From a numerical point of view, this property is desirable in a cutting plane or in a branch-and-cut algorithm.

Special cases and strengthening. We call flow-cutset inequalities with $A_2^- = \emptyset$ (or $E_2 = \emptyset$) *simple*. Simple flow-cutset inequalities can be strengthened by rounding down all left-hand side coefficients to the value of the right-hand side. Given that F_t is nondecreasing, this strengthening can either be applied to the base inequalities (6) and (7) or to the MIR inequalities (8) and (9). Both ways lead to the same strengthened simple flow-cutset inequalities (see [19] for details). We obtain

$$f^Q(\bar{A}_1^+) + \sum_{m \in M} F_t(\min(c^m, d_S^Q))x^m(A_1^+) \;\geq\; F_t(d_S^Q) \;\text{ and}$$
$$f^Q(\bar{A}_1^+) + \sum_{m \in M} F_t(\min(c^m, d_S^Q))x^m(E_1) \;\geq\; F_t(d_S^Q)\,.$$

For ease of exposition we assume that $c^m \leq d_S^Q$ for all $m \in M$ in the following discussion. Necessary and sufficient conditions for (8) and (9) to define facets of their corresponding polyhedra are derived in [2] and [19], respectively.

Inequalities (8) and (9) generalize large classes of known valid inequalities for network design polyhedra. In particular, by choosing $A_2^- = \emptyset$ ($E_2 = \emptyset$) and $A_1^+ = A_S^+$ ($E_1 = E_S$) as well as $Q = K_S^+$, they reduce to the well-known cutset inequalities that contain only module variables. Fixing the network cut and the module $t \in M$ leads to two cutset inequalities in the Directed case, one for each direction:

$$\sum_{m \in M} F_t(c^m)x^m(A_S^+) \;\geq\; F_t(d_S^{K_S^+}) \quad\text{and}\quad \sum_{m \in M} F_t(c^m)x^m(A_S^-) \;\geq\; F_t(|d_S^{K_S^-}|)\,.$$

The left-hand sides of the two cutset inequalities for the Bidirected and Undirected link types are identical. Taking the maximum of the two right-hand sides gives

$$\sum_{m \in M} F_t(c^m)x^m(E_S) \;\geq\; \max(F_t(d_S^{K_S^+}), F_t(|d_S^{K_S^-}|)).$$

Notice that we can assume $K_S^- = \emptyset$ for Undirected link models because every demand can be reversed without changing P_{un}. Hence every negative commodity, i. e., a commodity with $d_S^k < 0$, can be made positive with respect to a cut of the network. Reversing demand directions is done implicitly in our implementation.

Literature review. Cutset inequalities for Undirected models with up to three modules have been studied in a series of articles by Magnanti et al. [13, 14, 15] for the special case of divisible base capacities. Cutset inequalities for Bidirected models with two modules and divisible base capacities have been considered in Bienstock and Günlük [6]. These authors also introduce simple flow-cutset

inequalities. The general flow-cutset inequality (8) for DIRECTED models and a single module has been introduced by Chopra et al. [8]. The first one to study the general multi-module case and inequalities with arbitrary capacity structure is Atamtürk [2] in the context of the directed cutset polyhedron. A polyhedral study for network design polyhedra and all three link models is presented by Raack et al. [19]. General flow-cutset inequalities for a single module, a single commodity and BIDIRECTED models are extended by Rajan [20] to the concept of survivability using directed cycles and p-cycles.

3 Separation

Given any of the network design polyhedra P_{di}, P_{bi} and P_{un} and a (fractional) point $\hat{p} = (\hat{f}, \hat{x})$, the separation problem for general flow-cutset inequalities reduces to the problem of simultaneously determining a node-set $S \subset V$, a commodity subset $Q \subseteq K$, arc or edge subsets of the cut A_S or E_S and a module $t \in M$ leading to a most violated inequality. Atamtürk [2] shows that this problem is already $\mathcal{NP}$-hard in the special case of a single point-to-point commodity. By contrast, the separation of cutset inequalities and simple flow-cutset inequalities can be shown to be a max-flow problem if a single point-to-point commodity is assumed (see [2, 15]). For general commodity sets, finding a most violated cutset inequality is known to be $\mathcal{NP}$-hard (see Bienstock et al. [7]). For a fixed node-set S, the complexity of simultaneously determining Q and A_1^+, A_2^- (E_1, E_2) is not known. For fixed S and Q, however, suitable subsets of the cut arcs (cut edges) can be identified in linear time for every $t \in M$, as shown in [2] and used below.

The efficiency of separation routines based on flow-cutset inequalities for one of the three link types has been investigated by several authors. Some studies also incorporate the effect of arc residual capacity inequalities and three-partition inequalities. Except for Atamtürk [2] all studies consider divisible base capacities and a restricted module set with $|M| \leq 2$. Magnanti et al. [15] propose an enumeration strategy to generate cutset inequalities for the UNDIRECTED model with two modules. For small networks with $|V| \leq 12$ they consider all network cuts in an iterative process. If for $|S| = i - 1$ no violated cutset inequality can be found they check all cutset inequalities for violation with $|S| = i$ as long as $i \leq \lfloor |V|/2 \rfloor$. Considering a capacity formulation, Barahona [5] presents a separation routine for cutset inequalities that is based on a heuristic for the max-cut problem. Considering a BIDIRECTED model, Bienstock and Günlük [6] enumerate all node-sets S with the property that the two subgraphs $H[S]$ and $H[V \backslash S]$ defined by S are connected. This is done for two networks with $|V| \in \{15, 16\}$. Notice that this property is necessary for a cutset inequality to define a facet for the network design polyhedra P_{bi}, P_{un} (see [6, 13, 19]). Bienstock and Günlük check cutset inequalities, three-partition inequalities and simple flow-cutset inequalities in a hierarchical manner. If no cutset inequalities can be found they check three-partition inequalities and only if the latter fails they try to generate simple flow-cutset inequalities with $|Q| \leq 2$ and $|\bar{E}_1| \leq 3$. A similar approach was chosen by Bienstock et al. [7]. They study a DIRECTED model and enumerate a subset of all node-sets S with the property that $G[S]$ and $G[V \backslash S]$ are strongly connected for two networks with $|V| \in \{15, 27\}$. If the corresponding cutset inequality is violated or tight then also simple flow-cutset inequalities with small commodity and edge-sets are checked for violation. Bienstock et al. [7] introduce a second very fast heuristic to generate subsets S that is based on a shrinking procedure. A similar procedure is used by Günlük [9]. Atamtürk [2] reports computational results for network design problems with up to 29 nodes and 3 modules. Flow-cutset inequalities are considered for $|S| = |Q| = 1$ and tested against a pure branch-and-bound approach.

We distinguish three different approaches here. All three algorithms decompose the separation problem, combining ideas from [2, 6, 7, 9, 15]. Algorithm A heuristically computes a certain number of violated general flow-cutset inequalities. By contrast, Algorithm B exclusively generates cutset inequalities. Algorithm C follows a hierarchical approach that favors cutset inequalities over any other type of general flow-cutset inequalities. Recall that cutset inequalities form a subclass of general flow-cutset inequalities by taking $A_1^+ = A_S$ and $A_2^- = \emptyset$ (or $E_1 = E_S$ and $E_2 = \emptyset$). In the following, we provide a more detailed description of the algorithmic steps that all three separation routines are based on, and present some implementational details.

Algorithm A,B,C Generating General flow-cutset inequalities

Requires: LP-solution
Provides: violated general flow-cutset inequalities

1. Contract the graph. Enumerate in the resulting partition all cuts with their corresponding node-sets S.

2A. Proceed with Step 3.

2B. For every module $t \in M$ check the corresponding cutset inequality for violation. STOP.

2C. For every module $t \in M$ check the corresponding cutset inequality for violation. If no violated cutset inequalities can be found after several separation rounds proceed with Step 3 else STOP.

3. Given a node-set S, heuristically compute promising commodity subsets Q.

4. For given S and Q and for all $t \in M$, exactly determine subsets A_1^+, A_2^- or E_1, E_2 leading to a most violated (if any) general flow-cutset inequality of type (8) or (9). STOP.

Finding a node-set S To determine cutsets we generalize a heuristic proposed by Bienstock et al. [7] and Günlük [9]. Given weights w_a, w_e for the arcs and edges of the network, we iteratively shrink the arc (edge) with the largest weight, delete loops and maintain parallel links (i. e., the weights are not adapted during the shrinking process). We apply this procedure until the shrunken graph has a predefined number of nodes, and enumerate all cuts in that graph.

Given a fractional LP solution, let $s_a, s_e \geq 0$ be the slacks and $\pi_a, \pi_e \leq 0$ the dual values corresponding to the capacity constraints of arc a or edge e. For the BIDIRECTED link model there are two capacity constraints for every edge e, so we define slacks and duals as s_{e^+}, s_{e^-}, π_{e^+}, π_{e^-}. We define arc (edge) weights by

$$\begin{array}{lll} \text{DIRECTED link model:} & w_a := s_a + \pi_a & a \in A\,, \\ \text{BIDIRECTED link model:} & w_e := \min(s_{e^+}, s_{e^-}) + \min(\pi_{e^+}, \pi_{e^-}) & e \in E\,, \\ \text{UNDIRECTED link model:} & w_e := s_e + \pi_e & e \in E\,. \end{array}$$

With this definition, cuts are preferred that have many arcs (edges) with small slack to raise the chance of obtaining a violated general flow-cutset inequality. Since usually many capacity constraints have zero slack w. r. t. the current LP solution, we also consider the dual value as a second sorting criterion. Furthermore, increasing capacity on the cut shall maximally increase the objective function. In our computational tests, taking a combination of slacks and duals outperformed shrinking weights defined only by the slacks.

For the strength of cut-based inequalities it is crucial that the subgraphs defined by S and $V \backslash S$ are connected (BIDIRECTED and UNDIRECTED link types) or strongly connected (DIRECTED model), see Agarwal [1] and Raack et al. [19]. By definition of the shrinking procedure, every component of the resulting graph-partition is connected. Hence, if the shrinking procedure is continued until the shrunken graph has only two nodes left, then it can be guaranteed that the two subgraphs are connected. Notice however that even in this case it cannot be ensured that these subgraphs are strongly connected. Moreover, the separation routine turns out to be most effective if the shrunken graph has four to five nodes left and all cuts of the remaining graph are used to derive violated inequalities. This way more cuts are considered in each separation round, but it might happen that one of the subgraphs defined by a cut is not (strongly) connected, which results in weaker general flow-cutset inequalities. Although not incorporated in our algorithms, a strengthening is possible in this case by considering metric inequalities, see [9, 11] and the concluding remarks in [19].

In addition to the cuts obtained by shrinking we also consider all single node cuts, i. e., all node-sets $S = \{v\}$ with $v \in V$, in every separation round.

Finding a commodity subset Q In general no efficient way is known to compute a commodity subset that leads to a most violated general flow-cutset inequality. We concentrate on commodity subsets Q with $Q \subseteq K_S^+$ or, by symmetry, $Q \subseteq K_S^-$ (exchanging S and $V \backslash S$). Similar to [2, 6, 7], we use all singleton commodity subsets, some commodity subsets Q with $|Q| = 2$, and the whole set K_S^+ (K_S^-).

Finding subsets A_1^+ and A_2^- (or E_1 and E_2) Given a point $\hat{p} = (\hat{f}, \hat{x})$, the sets $S \subset V$, $Q \subseteq K$ and $t \in M$, we derive a most violated general flow-cutset inequality for the DIRECTED link model [2] in linear time by defining

$$A_1^+ := \Big\{ a \in A_S^+ : \sum_{m \in M} F_t(c^m) \hat{x}_a^m \leq \hat{f}_a^Q \Big\}, \quad A_2^- := \Big\{ a \in A_S^- : \sum_{m \in M} (c^m + F_t(-c^m)) \hat{x}_a^m < \hat{f}_a^Q \Big\}.$$

Similarly, for the BIDIRECTED and UNDIRECTED link models, we define

$$E_1 := \Big\{ e \in E_S : \sum_{m \in M} F_t(c^m) \hat{x}_e^m \leq \hat{f}_{e^+}^Q \Big\}, \quad E_2 := \Big\{ e \in E_S : \sum_{m \in M} (c^m + F_t(-c^m)) \hat{x}_e^m < \hat{f}_{e^-}^Q \Big\}.$$

This definition of the arc or edge-sets yields a general flow-cutset inequalities with minimal left-hand side. It is not the only possible definition with this property. If for instance for $a \in A_S^+$ the equality $\sum_{m \in M} F_t(c^m) \hat{x}_a^m = \hat{f}_a^Q$ holds, then assigning arc a to either of the sets A_1^+ or $\bar{A}_1^+$ leads to different general flow-cutset inequalities with the same absolute violation with respect to $\hat{p}$. It follows that ties can be broken arbitrarily, influencing the distribution of the generated inequalities (see Figure 6 and 8). The above definition turned out to be the most effective one in our computational tests. It implicitly favors the generation of cutset inequalities because we avoid generating flow-cutset inequalities with $\bar{A}_1^+ \neq \emptyset$ or $A_2^- \neq \emptyset$ ($\bar{E}_1 \neq \emptyset$ or $E_2 \neq \emptyset$).

The separation algorithms are implemented as callbacks using the callable library of CPLEX 10.0 [10]. The number of violated inequalities identified is enormous for most of the instances. Adding them all leads to unacceptable computation times since user-added cutting planes are never deleted in CPLEX. We used several techniques to select a small number of the most promising cuts. First, the total number of generated general flow-cutset inequalities is restricted to the number of rows in the initial formulation. Second, the algorithms are called only at the root node and at every 8th depth of the search tree. Third, every violated general flow-cutset inequalities found is written into a cut-pool. From all inequalities in the pool we only add a small number to the current formulation in every iteration, preferring those that have a large distance to the point $\hat{p}$ and that are not too orthogonal to the objective function. If the number of violated inequalities that we select in this way is very large or very small, we adapt the efficiency measures accordingly. We also avoid adding cutting planes that are almost parallel to each other. The necessary fine-tuning of algorithmic parameters has been carried out in a series of computational tests. The results we present here are those obtained with our best settings.

4 Computational results

Instances. We tested the branch-and-cut approach using Algorithms A, B and C on all instances of the Survivable Network Design Library (SNDlib) [18] with the following model specifications: link models DIRECTED, BIDIRECTED or UNDIRECTED, capacity structure MODULAR or EXPLICIT, no fixed-charge costs, continuous routing allowing all paths, no hop limits and no survivability. These are 54 problem instances in total, 4 of which are DIRECTED, 25 BIDIRECTED, and 25 UNDIRECTED. There are 27 instances with MODULAR and EXPLICIT link capacity structure, respectively. The number of nodes of the underlying networks ranges from 10 to 65, the number of links from 18 to 172, and the number of demands from 22 to 1869. The maximum number of modules is 11 for MODULAR capacities and 55 for EXPLICIT capacities. Notice that for BIDIRECTED models we calculated the routing cost of a link as the given cost parameter times the sum of the flows in both direction which conflicts with

the definition in [18] for the SNDlib where only the cost for the maximum of the two flow values is incurred.

We used Cplex 10.0 [10] in the default mode and compared its performance to our algorithms that augment Cplex by the presented separation routines. All computations were performed on a 3GHz x86 Linux machine with 2GB of memory and a time limit of one hour. For the statistics we distinguish between *easy* and *hard* instances. An instance is *easy* if it can be solved to optimality within the time limit of one hour by one of the considered algorithms. All other instances are referred to as being *hard*. These cannot be solved within the time limit either by default Cplex or by adding flow-cutset inequalities using Algorithms A, B or C. With respect to this definition, 14 Modular (13 Explicit) instances turned out to be easy, compared to 13 (respectively 14) hard instances.

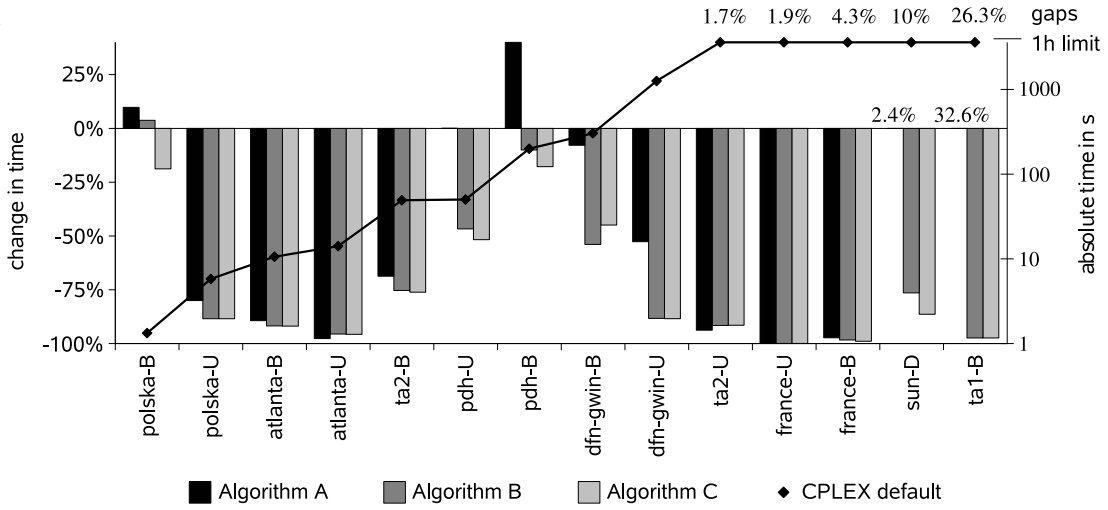


Figure 2: Easy instances – Modular: Relative change in solving time compared to default Cplex [bars], as well as absolute solving time of default Cplex [line with markers]. For instances unsolved within the time limit, the remaining optimality gap is reported.

Results for modular link capacities. We start with a detailed comparison of the results for the models with a Modular capacity structure on the links. Figure 2 reports on the acceleration compared to default Cplex for easy instances. The bars represent the change in the absolute CPU time. A value of -80% means that the solution time of default Cplex could be reduced by 80% using the respective algorithm. The black markers refer to the second y-axis, which displays the absolute solving times of default Cplex in seconds (on a logarithmic scale), according to which the instances are sorted. Those instances reaching the time limit with default Cplex are ordered according to the remaining optimality gap (i.e., $(upper - lower)/lower$, where *upper* and *lower* are the final primal and dual bound, respectively). For those instances, the acceleration is computed with the time limit as an underestimation of the real computation time of default Cplex. All instances are labeled with the name of the underlying network and the link model, where the letters U, B and D denote the Undirected, Bidirected and Directed model, respectively.

Algorithms B and C solve all easy instances to optimality within one hour of computation time. In contrast, Cplex fails to solve five of these instances in the default settings and two of these five are not solved by using Algorithm A. The remaining optimality gaps of Cplex and Algorithm A are reported in the upper part of Figure 2. For nearly all of the easy instances the computation time is drastically reduced by adding flow-cutset inequalities, in particular with Algorithms B and C. Even for the instances that cannot be solved by Cplex, the solution time is reduced to less than 10 minutes. Comparing Algorithms B and C, it turns out that for most of the instances it suffices to

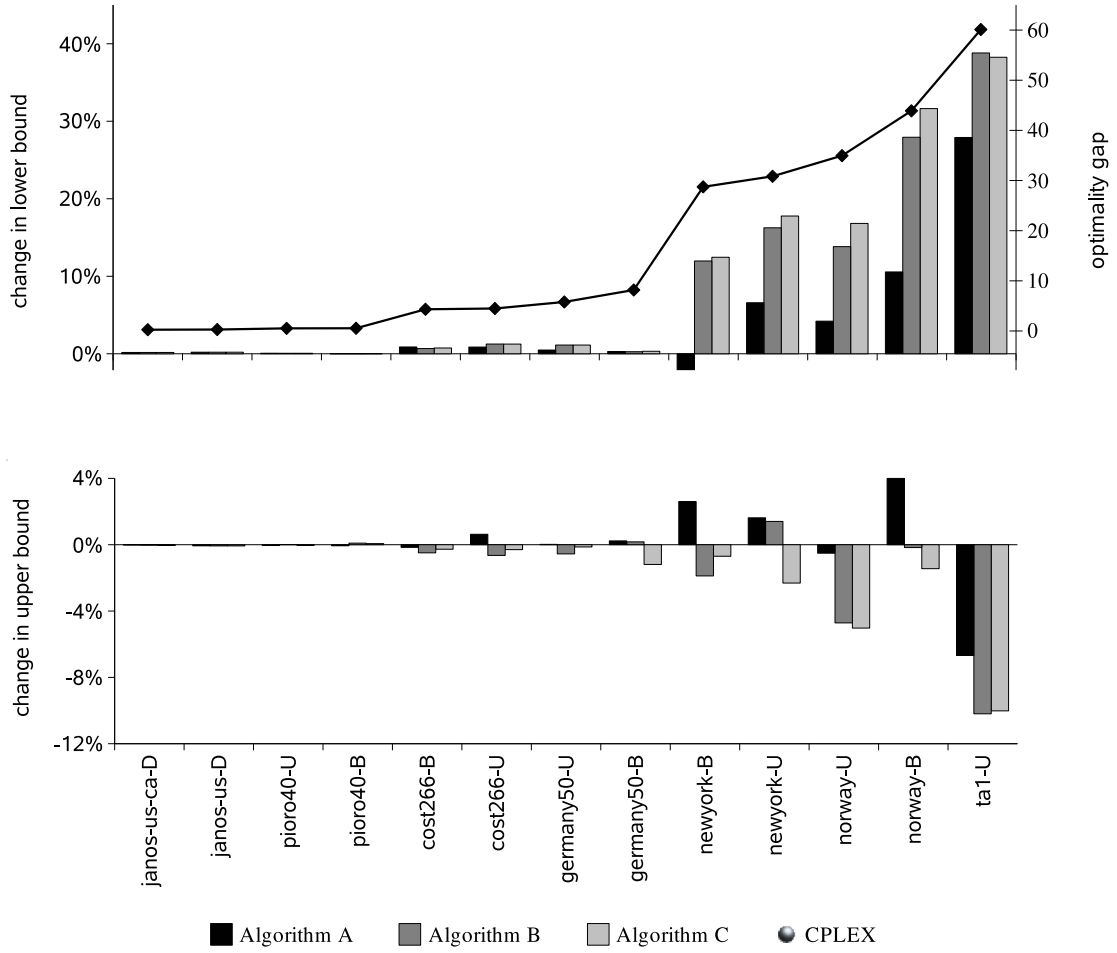


Figure 3: Hard instances – Modular: Relative change in lower and upper bound compared to default Cplex [bars], remaining optimality gap of default Cplex [line with markers]

add cutset inequalities to the initial formulation, but an additional speed-up could be obtained in particular cases by adding other types of general flow-cutset inequalities using Algorithm C.

For those Modular instances which are hard to solve, Figure 3 shows the relative change in the final lower and upper bounds. A value of 20% means that the corresponding final bound is 20% larger than the one obtained by default Cplex after one hour of computation. The instances are sorted by the remaining optimality gap of default Cplex, displayed on the second y-axis. Both lower and upper bounds are improved by general flow-cutset inequalities.

Again, Algorithms B and C outperform Algorithm A, and Algorithm C is slightly better than Algorithm B. It turns out that the improvement in the primal and dual bound is correlated to the remaining optimality gap of default Cplex. For the instance with the largest remaining optimality gap, ta1-U, the lower bound can be increased by almost 40% and the upper bound is decreased by more than 10%. Whenever Cplex has problems in finding cutting planes to increase the lower bound or in finding solutions to decrease the upper bound, the effect of adding general flow-cutset inequalities is significant. On the other hand, the performance of Cplex is not deteriorated by the cutting planes for those instances that have small gaps already in the default settings. The changes of the resulting optimality gaps are reported in Figure 4. Again the percentages are given with respect to the values obtained by Cplex in the default settings. It can be seen that the cutting planes most often significantly reduce the optimality gaps, sometimes by more than 90%.

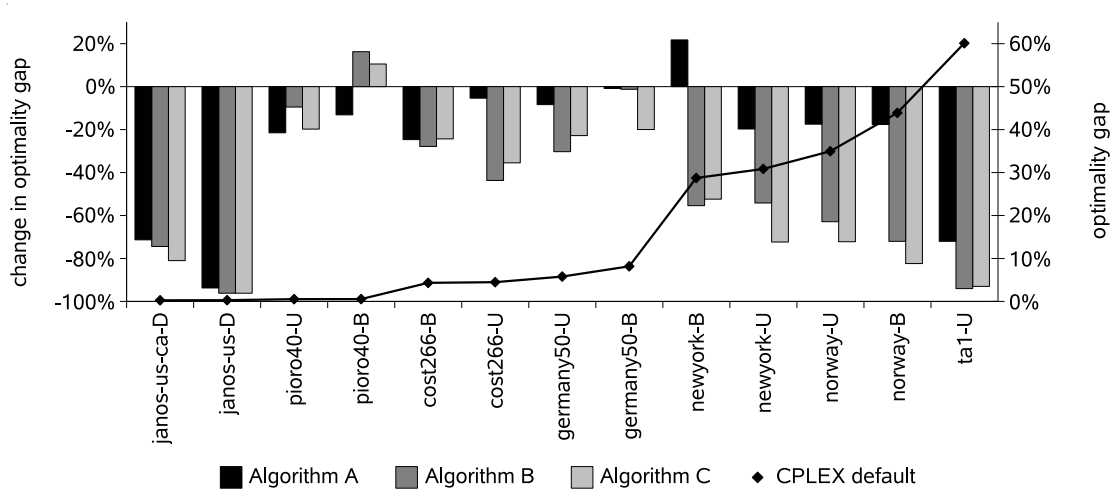
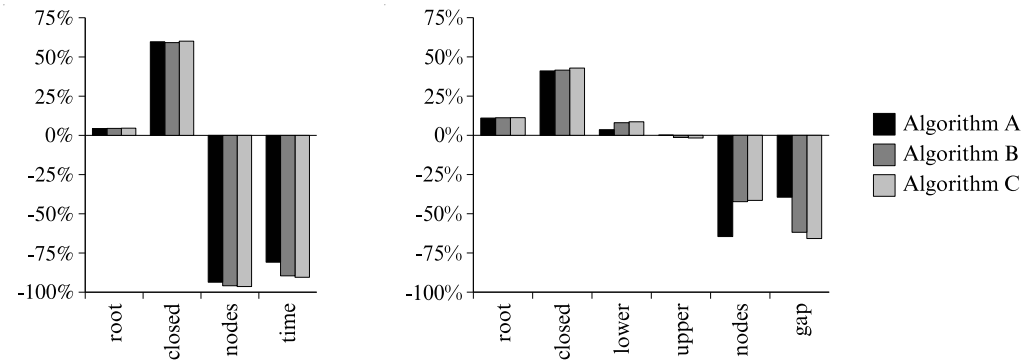


Figure 4: Hard instances – MODULAR: Relative change in the optimality gap compared to default CPLEX [bars], as well as the absolute optimality gap of default CPLEX [line with markers]



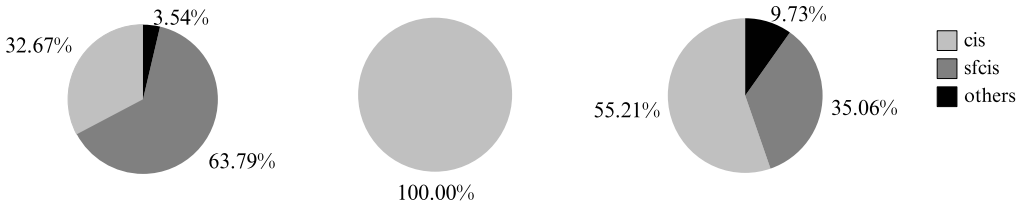
(a) Easy test instances

(b) Hard test instances

Figure 5: MODULAR: Relative changes of key measures on average when adding general flow-cutset inequalities compared to the values obtained by default CPLEX

Figure 5 summarizes the relative changes of several performance indicators when general flow-cutset inequalities are added, compared to CPLEX in the default settings. For this statistic we calculated the ratio of the considered measure with and without the respective separation routine and averaged these ratios over all instances using the geometric mean. The resulting mean value was then translated into a percentage. The figure shows the change in the the lower bound at the root node before branching (*root*), the solution time (*time*), the number of visited branch-and-bound nodes (*nodes*), the final lower and upper bounds (*lower, upper*), the final gap (*gap*), and the integrality gap closed at the root node (*closed*). The latter is defined as $(root - lp)/(best - lp)$, where lp is the value of the linear programming relaxation and *best* is the best known upper bound.

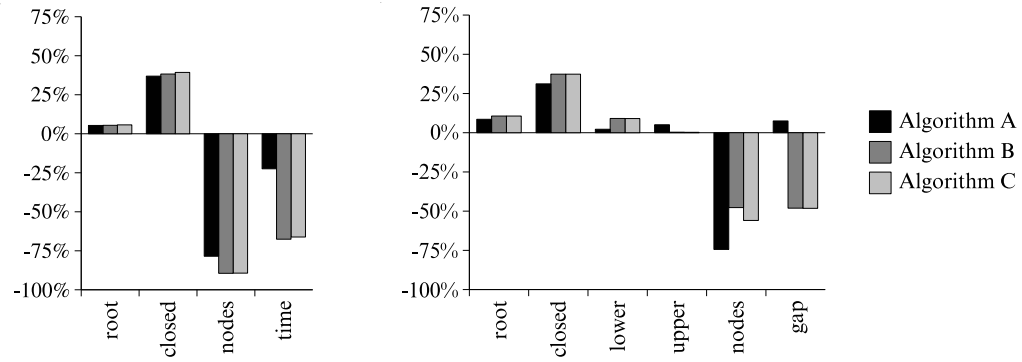
For easy instances the saved computation time correlates with the number of visited nodes and the improved lower bound at the root node. The computation time is reduced by more than 80% on average, and the number of nodes is even reduced by more than 90%. For hard instances, adding general flow-cutset inequalities leads to a decrease in the number of branch-and-bound nodes that can be visited within one hour of computation time. But since the cutting planes have a positive effect on both the lower and upper bound, the final gap is still reduced by more than 60% on average.



(a) Algorithm A (b) Algorithm B (c) Algorithm C

Figure 6: MODULAR: Average distribution of general flow-cutset inequalities added by Algorithms A, B and C

Among all tested separation algorithms, the hierarchical approach (Algorithm C) performed best. To understand the different behavior of the three algorithms we examined the distribution of the generated inequalities. Figure 6 shows the percentage of cutset inequalities (*cis*), of simple flow-cutset inequalities (*sfcis*) and of all other types of general flow-cutset inequalities (*others*). With the integrated approach of Algorithm A, only 32.67% of all generated general flow-cutset inequalities are cutset inequalities, whereas 100% and 55.21% are cutset inequalities with Algorithms B and C, respectively. Given that Algorithm C only slightly increases the performance compared to Algorithm B, this means that cutset inequalities are responsible for most of the progress. All other types of general flow-cutset inequalities should carefully be generated and only if no violated cutset inequalities are found. As already mentioned in Section 3, we restricted the total number of added general flow-cutset inequalities to the number of rows in the initial formulation. With this restriction, the number of inequalities added amounted to 62%, 44% and 48% of the size of the initial formulation for Algorithms A, B and C, respectively, averaged over all MODULAR instances.



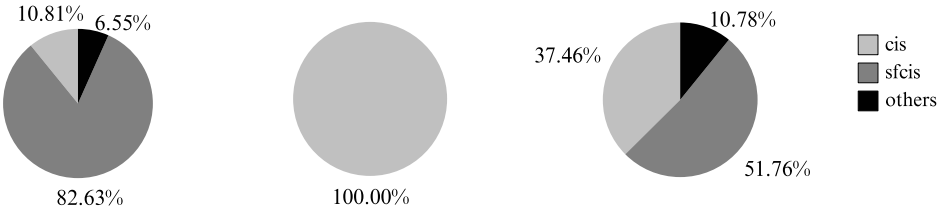
(a) Easy test instances (b) Hard test instances

Figure 7: EXPLICIT: Relative changes of key measures on average when adding general flow-cutset inequalities compared to the values obtained by default CPLEX

Results for explicit link capacities. We conclude this section with a summary of the results for models with an EXPLICIT link capacity structure. Remember that the EXPLICIT formulation is obtained by adding a general upper bound constraint for every arc or edge to the MODULAR formulation. Since the presented general flow-cutset inequalities neither exploit these additional inequalities nor the implied bounds on the capacity variables, we had expected a significant decline in the benefit of the three separation routines compared in the MODULAR case. But as the following

results show, general flow-cutset inequalities are of significant practical usefulness even for models with binary capacity variables. There are 13 easy and 14 hard to solve Explicit instances. Cplex in its default settings could not solve five of the easy instances within the time or memory limit. Algorithm A stills fails to solve two of the easy instances, whereas Algorithms B and C solve all easy instances to optimality within the time limit.

Figure 7 reports on the average performance of the three algorithms. The figure shows that the positive effect on the lower bound, the computation time (easy instances) and the optimality gap (hard instances) is still large but less than in the Modular case. Again, Algorithms B and C outperform Algorithm A. The computation time for easy instances is reduced by about 66% on average (compared to 80% in the Modular case). The remaining gap for hard to solve instances can still be reduced by 48% (compared to 66% in the modular case). In contrast to the Modular case, we observed a small negative effect on the upper bound, which is, however, largely outweighed by the increased lower bound.



(a) Algorithm A (b) Algorithm B (c) Algorithm C

Figure 8: Explicit: Average distribution of general flow-cutset inequalities added by Algorithms A, B and C

The distribution of the general flow-cutset inequalities that are added to the initial formulation is shown in Figure 8. It turns out that for Algorithms A and C the percentage of violated cutset inequalities is smaller than in the Modular case, and therefore other types of general flow-cutset inequalities are generated more often.

Due to the larger number of modules for the Explicit instances in SNDlib, the total number of added inequalities increases. Compared to the number of rows in the initial problem formulation, a percentage of 91%, 58% and 63% general flow-cutset inequalities was added by Algorithms A, B and C, respectively, averaged over all Explicit instances. With the large number of modules also the separation time, i. e., the time consumed by the separation routines becomes critical. For two of the hard instances (nobel-eu and nobel-us, both with 40 link modules) the separation time amounted to more than half of the total CPU time although we restricted the number of considered modules in step 2 and 4 of the Algorithms A, B and C to 10. Notice that the number of modules not only influences the number of iterations but also the support of the added inequalities and hence the time consumed by frequently called operations like the computation of the relative violation or checks for orthogonality. Nevertheless, the described separation routines turned out to be stable and useful for the case of Explicit link capacities.

5 Conclusion

In this paper we have studied capacitated network design problems with Directed, Bidirected, and Undirected link models, as well as Modular and Explicit link capacities with an arbitrary number of modules on the network links. The corresponding link-flow formulations are the basis of many real-world optimization problems, in particular in the field of telecommunications. We have focused on the class of flow-cutset inequalities to improve the dual bound, and showed how this class can be generalized to the three link types and arbitrarily many modules. We have discussed three

different generic separation algorithms and implementational details. These algorithms were used as separators within the branch-and-cut-solver CPLEX and tested against CPLEX with default settings on 54 network design problems stemming from the Survivable Network Design Library (SNDlib).

It was shown that the considered separation algorithms can drastically improve the performance of CPLEX for the given instances. The best results were obtained by using an hierarchical approach that favors cutset inequalities over other types of general flow-cutset inequalities. It turns out that this subclass of cutting planes, which contains only capacity variables but no flow variables, is responsible for most of the progress. The performance is only slightly better and varies over the instances when the larger class of general flow-cutset inequalities is considered. The latter ones should be added carefully and only if no violated cutset inequalities can be found. For almost all of the instances we had no difficulties in finding violated general flow-cutset inequalities, but the challenge is to find the best ones. We believe that a more elaborate selection of cuts and in particular of commodity subsets might even increase the impact of these inequalities. The results suggest that the success of those flow-cutset inequalities which are not cutset inequalities is rather problem-specific. It might depend on parameters such as the underlying network, flow-cost and the capacity structure. Also the MIR and flow-cover inequalities generated by CPLEX might influence the impact of flow-cutset inequalities. These dependencies have not been studied in this article.

The excellent results for models with EXPLICIT link capacities are unexpected, because general flow-cutset inequalities do not exploit the fact that the capacity variables are binary. The gain in the performance is not as large as for MODULAR models but still striking. This motivates the use of general flow-cutset inequalities as valid inequalities for single node flow sets in addition to flow-cover inequalities.

References

[1] Y. K. Agarwal. k-partition-based facets of the network design problem. *Networks*, 47(3):123–139, 2006.

[2] A. Atamtürk. On capacitated network design cut-set polyhedra. *Mathematical Programming*, 92:425–437, 2002.

[3] A. Atamtürk. On the facets of the mixed-integer knapsack polyhedron. *Mathematical Programming*, 98:145–175, 2003.

[4] A. Atamtürk. Sequence independent lifting for mixed-integer programming. *Operations Research*, 52:487–490, 2004.

[5] F. Barahona. Network design using cut inequalities. *SIAM Journal on Optimization*, 6:823–837, 1996.

[6] D. Bienstock and O. Günlük. Capacitated network design – polyhedral structure and computation. *INFORMS Journal on Computing*, 8:243–259, 1996.

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

[8] S. Chopra, I. Gilboa, and S. T. Sastry. Source sink flows with capacity installation in batches. *Discrete Applied Mathematics*, 86:165–192, 1998.

[9] O. Günlük. A branch and cut algorithm for capacitated network design problems. *Mathematical Programming*, 86:17–39, 1999.

[10] CPLEX 10.0. ILOG CPLEX Division, 889 Alder Avenue, Suite 200, Incline Village, NV 89451, USA. `http://www.ilog.com/products/cplex/`.

[11] M. Iri. On an extension of the maximum-flow minimum cut theorem to multicommodity flows. *Journal of the Operations Research Society of Japan*, 13(3):129–135, 1971.

[12] Q. Louveaux and L. A. Wolsey. Lifting, superadditivity, mixed integer rounding and single node flow sets revisited. *4OR*, 1(3):173–207, 2003.

[13] T. L. Magnanti and P. Mirchandani. Shortest paths, single origin-destination network design and associated polyhedra. *Networks*, 33:103–121, 1993.

[14] T. L. Magnanti, P. Mirchandani, and R. Vachani. The convex hull of two core capacitated network design problems. *Mathematical Programming*, 60:233–250, 1993.

[15] T. L. Magnanti, P. Mirchandani, and R. Vachani. Modelling and solving the two-facility capacitated network loading problem. *Operations Research*, 43:142–157, 1995.

[16] H. Marchand and L. A. Wolsey. Aggregation and mixed integer rounding to solve MIPs. *Operations Research*, 49(3):363–371, 2001.

[17] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988.

[18] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. ZIB Report 07-15, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2007. `http://sndlib.zib.de`.

[19] C. Raack, A.M.C.A. Koster, and R. Wessäly. On the strength of cut-based inequalities for capacitated network design polyhedra. ZIB Report 07-08, Konrad-Zuse-Zentrum für Informationstechnik Berlin, June 2007.

[20] D. Rajan. *Designing capacitated survivable networks: Polyhedral analysis and algorithms*. PhD thesis, University of California, Berkeley, 2004.