Commodity Representations and Cutset-Based Inequalities for Multicommodity Capacitated Fixed-Charge Network Design

Mervat Chouman
College of Business
Effat University
e.mail: mchuman@effatuniversity.edu.sa

Teodor Gabriel Crainic
Département de management et technologie
École des sciences de la gestion, UQAM
and
Interuniversity Research Centre on
Enterprise Networks, Logistics and Transportation (CIRRELT)
e.mail: TeodorGabriel.Crainic@cirrelt.ca

Bernard Gendron
Département d'informatique et de recherche opérationnelle
Université de Montréal
and
Interuniversity Research Centre on
Enterprise Networks, Logistics and Transportation (CIRRELT)
e.mail: Bernard.Gendron@cirrelt.ca

October 6, 2015

Abstract

We improve the mixed-integer programming formulation of the multicommodity capacitated fixed-charge network design problem by incorporating valid inequalities into a cutting-plane algorithm. We use five classes of known valid inequalities: the strong, cover, minimum cardinality, flow cover, and flow pack inequalities. The first class is particularly useful when a disaggregated representation of the commodities is chosen, while the last four are expressed in terms of network cutsets. We develop efficient separation and lifting procedures for these classes of inequalities. We present computational results on a large set of instances of various characteristics, allowing us to measure the impact of the different classes of valid inequalities on the quality of the lower bounds, in particular with respect to the representation of the commodities.

Key words : Multicommodity capacitated fixed-charge network design, commodity representation, cutset-based inequalities, separation, lifting.

1 Introduction

Network design models are used in many application areas, most notably in transportation and logistics [19, 45, 49]. These models span the entire spectrum of planning levels. At the strategic level, typical decisions involve the construction of infrastructures, the location of facilities and the acquisition of assets, taking into account long-term demands for product movements and vehicle flows [14, 60]. At the tactical level, decisions are often related to the selection of service routes by carriers, and the frequencies and schedules of these routes; such *service network design* problems arise in, e.g., maritime [13], rail [15, 62], and intermodal [24] transportation. At the operational level, service routes must be established on a short-term horizon, typically one day; examples include express shipment services [4], *adaptive distribution systems*, where facilities (such as parking spaces) are used or not according to demand fluctuations [30], and applications in *city logistics*, which involve network design and vehicle routing decisions [25].

In this paper, we study the multicommodity capacitated fixed-charge network design problem ($MCND$), a generic problem that captures many salient features of network design applications encountered in transportation and logistics. Given a directed graph $G = (N, A)$, where N is the set of nodes and A is the set of arcs, and a set of commodities K to be routed according to a known demand $d^k > 0$ flowing from an origin $O(k)$ to a destination $D(k)$ for each commodity k, the problem is to satisfy the demand at minimum cost. The objective function consists of the sum of transportation costs and fixed design costs, the latter being charged whenever an arc is used. The transportation cost on arc (i, j) is denoted $c_{ij} \geq 0$, while the fixed design cost for arc (i, j) is denoted $f_{ij} \geq 0$. In addition, there is a capacity $u_{ij} > 0$ on the flow of all commodities on arc (i, j); we assume $u_{ij} \leq \sum_{k \in K} d^k$ for each arc (i, j). The $MCND$ is NP-hard since it contains as a special case the multicommodity uncapacitated fixed-charge network design problem (obtained by imposing $u_{ij} = \sum_{k \in K} d^k$ for all $(i, j) \in A$), which is NP-hard [45].

The $MCND$ can be modeled as a mixed-integer program (MIP) by using continuous flow variables x_{ij}^k, which reflect the amount of flow on each arc (i, j) for each commodity k, and 0-1 design variables y_{ij}, which indicate if arc (i, j) is used or not:

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij}, \tag{1}$$

$$\sum_{j \in N_i^+} x_{ij}^k - \sum_{j \in N_i^-} x_{ji}^k = \begin{cases} d^k, & \text{if } i = O(k), \\ -d^k, & \text{if } i = D(k), \\ 0, & \text{otherwise,} \end{cases} \quad i \in N,\ k \in K, \tag{2}$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij}, \quad (i, j) \in A, \tag{3}$$

$$x_{ij}^k \geq 0, \quad (i, j) \in A,\ k \in K, \tag{4}$$

$$0 \leq y_{ij} \leq 1, \quad (i, j) \in A, \tag{5}$$

$$y_{ij} \quad \textit{integer} \quad (i, j) \in A, \tag{6}$$

where $N_i^+ = \{j \in N | (i,j) \in A\}$, $N_i^- = \{j \in N | (j,i) \in A\}$. Constraints (2) correspond to flow conservation equations for each node and commodity. Relations (3) represent capacity constraints for each arc. They also link together flow and design variables by forbidding flow to use an arc that is not chosen as part of the design.

Branch-and-bound (B&B) algorithms based on linear programming (LP) relaxations are the most common methods to solve such models. Here, however, the LP relaxation generally provides weak lower bounds [20]. Alternative relaxation approaches have been devised, in particular Benders decomposition [17] and Lagrangian-based procedures [20, 21, 29, 38, 40, 57]. Heuristic methods have also been proposed for computing feasible solutions [18, 22, 23, 31, 32, 37, 39, 56]. In this paper, we present a cutting-plane method for improving the LP relaxation lower bounds. Although this methodology has been applied successfully to other, closely related, network design problems [1, 2, 6, 8, 10, 11, 12, 27, 28, 41, 43, 44, 52, 54], it has not been used to address the $MCND$. Our objective is to identify inequalities that can be useful within a cutting-plane framework by exploiting simple structures derived from relaxations of the $MCND$. We aim to perform an extensive computational study of the impact of these inequalities on improving the lower bounds for a large set of instances used in prior works on the $MCND$.

The cutting-plane method we propose is based on five classes of known valid inequalities (VI): the strong, cover, minimum cardinality, flow cover, and flow pack inequalities. The last four classes of inequalities are expressed in terms of cutsets, where the set of nodes is partitioned into two subsets. The five classes of inequalities are derived from three relaxations of the $MCND$: the single-arc design relaxation (for strong inequalities), the single-cutset relaxation (for cover and minimum cardinality inequalities), and the single-cutset flow relaxation (for flow cover and flow pack inequalities). These relaxations display problem structures for which the VI we use are facet-defining under mild conditions. We recall these known results in Section 2.

Other classes of VI can be derived. For instance, instead of using single cutset structures, one might use collections of cutsets in a single inequality [46, 55]. Another idea is to partition the set of nodes into k subsets with $k \geq 2$ [3, 12, 28, 43, 44]. Such inequalities can also be combined together to derive other VI by applying mixed-integer rounding [7, 12, 28, 47, 54]. Although these ideas for generating other VI have proven effective for some related problems, especially those involving general integer variables, our choice of inequalities is based on the abundant literature that demonstrates the strength of cover and flow cover inequalities for mixed 0-1 programs and the impact of VI based on cutsets for strengthening network design MIP models.

A key to the success of these inequalities is the representation of the commodities: it is well-known in multicommodity network flow problems that commodities sharing the same origin or the same destination can be aggregated into a single commodity. This transformation provides the same sets of feasible and optimal solutions than the original, or *disaggregated*, commodity representation, whenever there are no commodity-dependent costs or capacities, which is the case for the $MCND$. In this paper, we explore the results obtained with an alternative commodity representation that aggregates all commodities with the same origin, called *aggregated* commodity representation.

Cutting-plane approaches for related multicommodity network design problems, which have all used cutsets to derive classes of VI, have chosen either the aggregated [11, 12, 28, 54] or the disaggregated commodity representation [10, 27, 44]. To the best of our knowledge, there was no attempt to look at the effect of commodity representation on the strength of the LP relaxations obtained by adding cutset-based inequalities. We attempt to fill this gap. The advantage of the disaggregated commodity representation is that some VI can be stronger in that case (for example, the strong inequalities), while the advantage of the aggregated commodity representation is the reduction in the size of the model (the number of flow variables being reduced by a factor of $|V|$). Although the combination of disaggregated commodity representation and strong inequalities provides strong lower bounds [20], it might be preferable to use the weaker, but more compact, aggregated commodity representation in the hope that significant lower bound improvements are obtained by adding cutset-based inequalities.

To the best of our knowledge, the only known theoretical result on the relationships between commodity representation and cutset-based inequalities is for the case of single-commodity uncapacitated fixed-charge network design with one origin and multiple destinations [55]. In that special case of the $MCND$, one can derive an equivalent multicommodity formulation by associating a commodity to each destination. Rardin and Wolsey [55] show that the multicommodity LP relaxation enriched with strong inequalities is equivalent to a single-commodity LP relaxation strenghtened with so-called dicut collection inequalities, a class of VI derived from cutsets. As pointed out by these authors, no equivalent result is known for capacitated problems, even in the single-commodity case.

Our contribution is threefold. First, we develop a cutting-plane algorithm that includes separation and lifting procedures adapted to the $MCND$. In particular, we present a new separation procedure for flow cover and flow pack inequalities. We develop procedures for generating cutsets, including a method inspired by metaheuristics approaches, which can be adapted to other network design problems. Second, we perform computational experiments that show the efficiency of our separation, lifting and cutset generation methods. We show that our cutting-plane algorithm is competitive with that of the state-of-the-art MIP solver CPLEX (version 12) on a large set of instances. When embedded in the B&B algorithm of CPLEX, we show that our cutting-plane procedure allows to prove optimality for a majority of the instances, while the unsolved instances show an average optimality gap within 2% when stopped after a reasonable CPU time limit. Third, we compare, with our cutting-plane algorithm, the relative strength of the different classes of inequalities when using the aggregated and disaggregated commodity representations. We show that large-scale instances with many commodities (more than 100) perform best with the disaggregated commodity representation, while small-scale instances with few commodities (around 10) benefit from the aggregated commodity representation.

The paper is organized as follows. In Section 2, we describe the five classes of VI and the relaxations from which they are derived. The separation and lifting procedures for these inequalities are presented in Section 3. The cutting-plane algorithm, including the cutset generation procedure, is the topic of Section 4. In Section 5, we report the results of experiments on a large class of problem instances. We conclude this paper with a discussion of future research avenues.

2 Relaxations and Valid Inequalities

In this section, we present three relaxations of the $MCND$, the single-arc design, single-cutset, and single-cutset flow problems, from which we identify the five classes of VI that are used in our cutting-plane algorithm. We use the following notation: for any model MOD, its set of feasible solutions is denoted $F(MOD)$, while the convex hull of $F(MOD)$ is denoted $conv(F(MOD))$.

2.1 Single-Arc Design Relaxation and Strong Inequalities

We relax the flow conservation equations and replace them by inequalities (7), which are derived from the observation that any optimal solution is circuit-free, since all costs are nonnegative:

$$x_{ij}^k \leq d^k, \quad \forall (i,j) \in A, k \in K. \tag{7}$$

The resulting relaxation decomposes by arc; following the terminology in [43], we call the resulting problem associated with each arc (i,j) the *single-arc design relaxation*, SAD_{ij}. Its feasible set can be written as follows:

$$F(SAD_{ij}) = \{ (x_{ij}^k)_{k \in K}, y_{ij} \mid \sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij},\ 0 \leq x_{ij}^k \leq d^k, k \in K,\ y_{ij} \in \{0,1\} \}. \tag{8}$$

This set also arises when relaxing the demand constraints in the capacitated facility location problem ($CFLP$): an arc in the $MCND$ corresponds to a facility in the $CFLP$ and a commodity in the $MCND$ corresponds to a customer in the CFLP. The following *strong inequalities* (SI) are valid for $F(SAD_{ij})$:

$$x_{ij}^k \leq d^k y_{ij}, \quad \forall k \in K. \tag{9}$$

These inequalities are not only facet-defining for $conv(F(SAD_{ij}))$, but together with the other inequalities, they define the convex hull of solutions (a proof can be found in the study of Lagrangian relaxations for the $CFLP$ by Cornuéjols et al. [16]):

$$conv(F(SAD_{ij})) = \{ (x_{ij}^k)_{k \in K}, y_{ij} \mid \sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij},\ 0 \leq x_{ij}^k \leq d^k y_{ij}, k \in K,\ y_{ij} \in [0,1] \}. \tag{10}$$

Adding the SI for all arcs to the $MCND$ LP relaxation significantly improves the quality of the lower bound [20, 29]. Although there is a polynomial number of SI ($|A||K|$), adding all of them to the LP relaxation yields large models that frequently exhibit degeneracy. Only a small number of SI are added with our cutting-plane algorithm.

2.2 Single-Cutset Relaxation and Knapsack Inequalities

If we let $S \subset N$ be any non-empty subset of N and $\bar{S} = N \backslash S$ its complement, we denote the corresponding cutsets by $(S, \bar{S}) = \{(i,j) \in A \mid i \in S, j \in \bar{S}\}$ and $(\bar{S}, S) = \{(i,j) \in A \mid i \in \bar{S}, j \in S\}$, and their associated commodity subsets $K(S, \bar{S}) = \{k \in$

$K \mid O(k) \in S,\, D(k) \in \bar{S}\}$ and $K(\bar{S},S) = \{k \in K \mid O(k) \in \bar{S},\, D(k) \in S\}$. For any $L \subseteq K$, we also introduce the following notation: $x_{ij}^L = \sum_{k\in L} x_{ij}^k$ for any arc (i,j), $d_{(S,\bar{S})}^L = \sum_{k\in K(S,\bar{S})\cap L} d^k$ and $d_{(\bar{S},S)}^L = \sum_{k\in K(\bar{S},S)\cap L} d^k$. By summing the flow conservation equations (2) for all $i \in S$ and $k \in L$, we obtain

$$\sum_{(i,j)\in(S,\bar{S})} x_{ij}^L - \sum_{(j,i)\in(\bar{S},S)} x_{ji}^L = d_{(S,\bar{S})}^L - d_{(\bar{S},S)}^L. \tag{11}$$

Replacing L by $K(S,\bar{S})$ in equation (11) and using the inequalities $x_{ij}^{K(S,\bar{S})} \leq u_{ij}y_{ij}$ for $(i,j) \in (S,\bar{S})$ and $x_{ji}^{K(S,\bar{S})} \geq 0$ for $(j,i) \in (\bar{S},S)$, we obtain the *single-cutset relaxation*, SC_S, whose feasible set is defined as follows, where $d_{(S,\bar{S})} \equiv d_{(S,\bar{S})}^{K(S,\bar{S})}$ (note that $d_{(\bar{S},S)}^{K(S,\bar{S})} = 0$ by definition):

$$F(SC_S) = \{\ (y_{ij})_{(i,j)\in(S,\bar{S})} \mid \sum_{(i,j)\in(S,\bar{S})} u_{ij}y_{ij} \geq d_{(S,\bar{S})},\ y_{ij} \in \{0,1\}, (i,j) \in (S,\bar{S})\ \}. \tag{12}$$

The single-cutset inequality defining $F(SC_S)$ states that there should be enough capacity on the arcs of the cutset $(S,\bar{S})$ to satisfy the total demand that must flow from S to $\bar{S}$.

By complementing the y variables (replacing y_{ij} by $1 - y_{ij}$) in $F(CS_S)$, the single-cutset relaxation reduces to a 0-1 knapsack structure. The well-known cover inequalities for that structure [9, 36, 61] are based on the following definitions (for the sake of clarity, we adapt to $F(SC_S)$ the terminology related to the 0-1 knapsack structure): $C \subseteq (S,\bar{S})$ is a *cover* if the total capacity of the arcs in $(S,\bar{S})\backslash C$ does not cover the demand, i.e., $\sum_{(i,j)\in(S,\bar{S})\backslash C} u_{ij} < d_{(S,\bar{S})}$; moreover, a cover $C \subseteq (S,\bar{S})$ is *minimal* if it is sufficient to open any arc in C to cover the demand, i.e., $\sum_{(i,j)\in(S,\bar{S})\backslash C} u_{ij} + u_{pq} \geq d_{(S,\bar{S})}, \forall (p,q) \in C$. For every cover $C \subseteq (S,\bar{S})$, the *cover inequality* (CI)

$$\sum_{(i,j)\in C} y_{ij} \geq 1 \tag{13}$$

is valid for $F(SC_S)$. This inequality states that at least one arc from the cover C must be opened in order to meet the demand. If C is a minimal cover, we can apply a lifting procedure to derive a facet of $conv(F(CS_S))$ [9, 61].

In addition to the cover inequalities, we use the so-called minimum cardinality inequalities. Let the capacities of the arcs in $(S,\bar{S})$ be sorted in non-increasing order: $u_{a(t)} \geq u_{a(t+1)}$, where $a(t) \in (S,\bar{S})$, $t = 1,...,|(S,\bar{S})|$ $(u_{a(t+1)} = u_{a(t)})$. Define then $l_{(S,\bar{S})} = \max\{h \mid \sum_{t=1,...,h} u_{a(t)} < d_{(S,\bar{S})}\} + 1$, the least number of arcs in $(S,\bar{S})$ that must be used in every solution of $F(SC_S)$. We then derive the *minimum cardinality inequality* (MCI):

$$\sum_{(i,j)\in(S,\bar{S})} y_{ij} \geq l_{(S,\bar{S})}. \tag{14}$$

This inequality has been used to strengthen relaxation bounds for the 0-1 knapsack problem [48].

As discussed in Section 3.1, we use the two families of knapsack inequalities, CI and MCI, in the following context: initially, some y variables are fixed to either 0 or 1 (using the LP relaxation solution), then, a violated inequality is generated for the resulting restriction of $F(SC_S)$, and finally, a lifting procedure is applied to obtain a VI for $F(SC_S)$. Different variable fixing strategies are used for the two types of inequalities, which yields different restrictions of $F(SC_S)$. In this context, it is possible to obtain an MCI stronger than a CI, even though the MCI is in general weaker than the facet-defining minimal CI.

2.3 Single-Cutset Flow Relaxation and Flow Cover Inequalities

To derive the single-cutset flow relaxation, we use the same notation as in the previous section. In addition, for any arc (i,j) and any $L \subseteq K$, we define $b^L_{ij} = \min\{u_{ij}, \sum_{k\in L} d^k\}$, which is an upper bound on the flow of all commodities in L that can use arc (i,j). Using this bound and relaxing equation (11), we obtain the *single-cutset flow relaxation*, SCF^L_S, whose feasible set is defined as

$$F(SCF^L_S) = \{\ (x^L_{ij}, y_{ij})_{(i,j)\in(S,\bar S)\cup(\bar S,S)} \mid \sum_{(i,j)\in(S,\bar S)} x^L_{ij} - \sum_{(j,i)\in(\bar S,S)} x^L_{ji} \le d^L_{(S,\bar S)}, \tag{15}$$

$$0 \le x^L_{ij} \le b^L_{ij} y_{ij},\ y_{ij} \in \{0,1\}, (i,j) \in (S,\bar S)\cup(\bar S,S)\ \}. \tag{16}$$

This relaxation reduces to the single-node fixed-charge flow problem, introduced in [53], and studied by many authors, since it arises as a natural relaxation for general MIP models. In particular, two classes of inequalities have been derived for the single-node fixed-charge flow problem, the flow cover and flow pack inequalities, which we now describe for $F(SCF^L_S)$.

A flow cover (C_1, C_2) is defined by two sets $C_1 \subseteq (S,\bar S)$ and $C_2 \subseteq (\bar S, S)$ such that $\mu = \sum_{(i,j)\in C_1} b^L_{ij} - \sum_{(j,i)\in C_2} b^L_{ji} - d^L_{(S,\bar S)} > 0$. The *flow cover inequality* (FCI) is then defined as

$$\begin{aligned}\sum_{(i,j)\in C_1} (x^L_{ij} + (b^L_{ij} - \mu)^+(1 - y_{ij})) \ \le\ & \sum_{(j,i)\in D_2} \min\{b^L_{ji}, \mu\} y_{ji} + \sum_{(j,i)\in C_2} b^L_{ji} \\ & + d^L_{(S,\bar S)} + \sum_{(j,i)\in(\bar S,S)\setminus C_2\cup D_2} x^L_{ji},\end{aligned} \tag{17}$$

where $a^+ = \max\{0, a\}$ and $D_2 \subset (\bar S, S)\setminus C_2$. This inequality has been studied by several authors [35, 42, 53, 59] and is implemented in state-of-the-art MIP software tools.

Using the same notation as above, a flow pack (C_1, C_2) is defined by two sets $C_1 \subseteq (S,\bar S)$ and $C_2 \subseteq (\bar S, S)$ such that $\mu = \sum_{(i,j)\in C_1} b^L_{ij} - \sum_{(j,i)\in C_2} b^L_{ji} - d^L_{(S,\bar S)} < 0$. The *flow pack inequality* (FPI) is then defined [5, 58] as

$$\begin{aligned}\sum_{(i,j)\in C_1} x^L_{ij} + \sum_{(i,j)\in D_1} (x^L_{ij} - \min\{b^L_{ij}, -\mu\} y_{ij})) \ \le\ & - \sum_{(j,i)\in C_2} (b^L_{ji} + \mu)^+(1 - y_{ji}) + \\ & \sum_{(j,i)\in(\bar S,S)\setminus C_2} x^L_{ji} + \sum_{(i,j)\in C_1} b^L_{ij}.\end{aligned} \tag{18}$$

where $D_1 \subset (S, \bar{S}) \backslash C_1$. The FPI can be viewed as a flow cover inequality for the relaxation of $F(SCF_S^L)$ defined by the inequality $\sum_{(j,i)\in(\bar{S},S)} x_{ji}^L - \sum_{(i,j)\in(S,\bar{S})} x_{ij}^L - t_{(S,\bar{S})}^L \leq -d_{(S,\bar{S})}^L$, where $t_{(S,\bar{S})}^L$ is a slack variable. Under mild conditions, both the FCI and FPI can be lifted to obtain facet-defining inequalities for $conv(F(SCF_S^L))$ [5, 35].

3 Separation and Lifting Methods

In this section, we present separation and lifting procedures for each class of VI presented above. We first note that the separation of strong inequalities is trivial, as it suffices to scan each arc and each commodity to identify all violated inequalities. For all cutset-based inequalities, we assume a cutset $(S, \bar{S})$ is given (see Section 4 for a description of cutset generation procedures). We first present separation and lifting for CI and MCI, and then we explain how we generate FCI and FPI using a new separation routine for these classes of inequalities. In this section and in the remainder of the paper, we use $(\bar{x}, \bar{y})$, with the appropriate indices, to denote the current fractional LP solution.

3.1 Cover and Minimum Cardinality Inequalities

To generate cover and minimum cardinality inequalities, we first determine, *a priori*, two subsets C_1 (the open arcs) and C_0 (the closed arcs) in $(S, \bar{S})$ that satisfy the condition

$$\sum_{(i,j)\in(S,\bar{S})\backslash(C_1\cup C_0)} u_{ij} \geq d_{(S,\bar{S})} - \sum_{(i,j)\in C_1} u_{ij} > 0.$$

To find C_1 and C_0, we perform procedure **OpenCloseArcs** (summarized in Algorithm 1), which uses the variables U and D to represent, respectively, the residual capacity (i.e., $\sum_{(i,j)\in(S,\bar{S})\backslash(C_1\cup C_0)} u_{ij}$), and the residual demand (i.e., $d_{(S,\bar{S})} - \sum_{(i,j)\in C_1} u_{ij}$). The procedure makes use of the current LP solution $\bar{y}$, attempting to close an arc (i, j) with a small value $\bar{y}_{ij}$ (as measured by a threshold ϵ) and such that the residual capacity after closing arc (i, j) still covers the residual demand D (i.e., $U - u_{ij} \geq D$). Similarly, the procedure attempts to open an arc (i, j) with a large value $\bar{y}_{ij}$ (as measured by a threshold $1 - \epsilon$) and such that there is still some residual demand to cover after opening arc (i, j) (i.e., $D - u_{ij} > 0$). As in Gu *et al.* [33], the sets C_1 and C_0 can be derived from the variables having integer values at the current LP solution, by using ϵ arbitrarily close to 0.

We define the restricted single-cutset inequality induced by C_1 and C_0 as

$$\sum_{(i,j)\in(S,\bar{S})\backslash(C_1\cup C_0)} u_{ij} y_{ij} \geq d_{(S,\bar{S})} - \sum_{(i,j)\in C_1} u_{ij}.$$

To define a cover C for this restricted cutset inequality, we have implemented the heuristic approach proposed by Gu *et al.* [33, 34] in their extensive study of cover inequalities. The basic idea of this heuristic is to try to exclude as much as possible from the set C the arcs with large $\bar{y}_{ij}$, in order to increase the chance of finding a violated inequality

Algorithm 1 OpenCloseArcs

$U \leftarrow \sum_{(i,j)\in(S,\bar{S})} u_{ij}$, $D \leftarrow d_{(S,\bar{S})}$
for arc $(i,j) \in (S,\bar{S})$ (in arbitrary order) **do**
  if $(\bar{y}_{ij} \leq \epsilon)$ and $(U - u_{ij} \geq D)$ **then**
    Add (i,j) to C_0
    Close (i,j) by setting $U \leftarrow U - u_{ij}$
  end if
  if $(\bar{y}_{ij} \geq 1 - \epsilon)$ and $(D - u_{ij} > 0)$ **then**
    Add (i,j) to C_1
    Open (i,j) by setting $D \leftarrow D - u_{ij}$ and $U \leftarrow U - u_{ij}$
  end if
end for

(i.e., $\sum_{(i,j)\in C} \bar{y}_{ij} < 1$). Therefore, the heuristic considers the arcs in non-decreasing order of $\bar{y}_{ij}$, instead of $\frac{\bar{y}_{ij}}{u_{ij}}$, as would be performed by the classical greedy heuristic for the 0-1 knapsack problem. Ties are broken by considering the arcs in non-increasing order of their capacity. Once a cover is obtained, it is easy to extract a minimal cover from it, by removing some of the arcs until the cover becomes minimal. Once the cover C is constructed, the induced inequality might be strengthened by the lifting procedure presented next. Note that, even if the identified cover inequality is not violated, we might find a violated one through the lifting procedure.

To generate an MCI, it suffices to use a sorting algorithm to compute the least number of arcs that must be opened in the set $(S,\bar{S})\backslash(C_1 \cup C_0)$. Although the MCI is weak in general, by deriving it over a restriction of $(S,\bar{S})$, followed by the application of a lifting procedure, one can obtain a strengthened VI.

CI and MCI derived from the restricted cutset inequality have the following general form:

$$\sum_{(i,j)\in B} y_{ij} \geq L,$$

with $L = 1$ and B corresponding to a cover, in the case of a cover inequality, while for a minimum cardinality inequality, $B = (S,\bar{S})\backslash(C_1 \cup C_0)$ and L is equal to the least number of arcs that must be used in B. Since this inequality is restricted to open arcs in C_1 and closed arcs in C_0, lifting (down for the variables in C_1 and up for the variables in C_0) is necessary to ensure its validity for $F(SC_S)$.

Lifting amounts to determining coefficients γ_{ij} for all $(i,j) \in (S,\bar{S})\backslash B$ such that

$$\sum_{(i,j)\in(S,\bar{S})\backslash B} \gamma_{ij} y_{ij} + \sum_{(i,j)\in B} y_{ij} \geq L + \sum_{(i,j)\in(S,\bar{S})\backslash(B\cup C_0)} \gamma_{ij}$$

is valid for $F(SC_S)$. The lifting procedure is applied sequentially, meaning that the variables are lifted one after the other in some predetermined order. For each (i,j), it is well-known that the corresponding lifting coefficient γ_{ij} can be determined by solving a 0-1 knapsack problem. The quality of the resulting lifted inequality depends on the order

in which the variables are lifted. Note that, lifting down the variables in $(S, \bar{S})\backslash(B \cup C_0)$ contributes to the violation of the inequality since $\gamma_{ij} y_{ij} \leq \gamma_{ij}$. However, lifting up the variables in C_0 has a negative impact on the violation in the sense that an inequality violated prior to this lifting step might become satisfied after. This might happen if some variables in C_0 have positive values ($\bar{y}_{ij} > 0$) at the current LP solution. We conclude that lifting down the variables in $(S, \bar{S})\backslash(B \cup C_0)$ must be accomplished before lifting up the variables in C_0. Moreover, when lifting down the variables in $(S, \bar{S})\backslash(B \cup C_0)$, those with fractional values are lifted first, in non-decreasing order of their current values. Ties are broken by considering first the arcs in non-increasing order of their capacity. When lifting up the variables in C_0, we do the exact opposite.

The cover and minimum cardinality inequalities display similar structures and, thus, the same lifting strategy is used for both. Different values of the parameter ϵ in procedure **OpenCloseArcs** are used to define the restricted sets C_0 and C_1. For the CI, we set $\epsilon = 0$, i.e., all variables with an integer value are fixed to that value, as in [33]. For the MCI, following preliminary computational experiments, we set $\epsilon = 0.5$, which is somewhat intuitive. Indeed, unlike the CI, which is based on a minimal cover, the MCI by itself is not strong. Therefore, closing and opening as many arcs as possible, as reflected by the value $\epsilon = 0.5$, and then lifting the variables that have been fixed, will lead to a stronger inequality.

3.2 Flow Cover and Flow Pack Inequalities

To generate flow cover and flow pack inequalities, we use two simpler VI for $F(SCF_S^L)$. The first one is the *single-arc flow pack inequality* (SFPI), defined as:

$$\sum_{(i,j)\in C_1'} x_{ij}^L + x_{rt}^L \leq (\sum_{(j,i)\in C_2'} b_{ji}^L + d_{(S,\bar{S})}^L) y_{rt} + \sum_{(j,i)\in(\bar{S},S)\backslash C_2'} x_{ji}^L + (1 - y_{rt}) \sum_{(i,j)\in C_1'} b_{ij}^L, \quad (19)$$

where $(r,t) \in (S, \bar{S})$, $C_1' \subseteq (S, \bar{S})\backslash\{(r,t)\}$ and $C_2' \subseteq (\bar{S}, S)$. The second VI is called the *single-arc flow cover inequality* (SFCI):

$$\sum_{(i,j)\in C_1'} x_{ij}^L + x_{rt}^L \leq (\sum_{(j,i)\in C_2'} b_{ji}^L + d_{(S,\bar{S})}^L)(1 - y_{rt}) + \sum_{(j,i)\in(\bar{S},S)\backslash C_2'} x_{ji}^L + y_{rt} \sum_{(i,j)\in C_1'} b_{ij}^L, \quad (20)$$

where $(r,t) \in (\bar{S}, S)$, $C_1' \subseteq (S, \bar{S})$ and $C_2' \subseteq (\bar{S}, S)\backslash\{(r,t)\}$. The Appendix shows the validity of these inequalities and specifies conditions under which they can be used to derive violated FCI and FPI.

The interest of these single-arc inequalities is that their separation problems are simple, in contrast with the FCI and the FPI, which are hard to separate. Indeed, given $(\bar{x}, \bar{y})$ the current LP solution and an arc $(r,t) \in (S, \bar{S})$, separating the SFPI consists in setting

$$C_1' = \{(i,j) \in (S, \bar{S})\backslash\{(r,t)\} | \bar{x}_{ij}^L > (1 - \bar{y}_{rt}) b_{ij}^L\},$$
$$C_2' = \{(j,i) \in (\bar{S}, S) | b_{ji}^L \bar{y}_{rt} < \bar{x}_{ji}^L\}.$$

For each subset S generated by the cutting-plane algorithm, the separation procedure thus scans each arc in $(S, \bar{S})$, trying to find a violated SFPI associated with this arc. If S consists of a singleton containing the origin of commodity k, we set $L = \{k\}$ and $C'_2 = \emptyset$, since in this case there is no flow of commodity k coming into r. Otherwise, we set $L = \{k \in K | \bar{x}^k_{rt} > 0\}$, in order to maximize the left-hand side of (19) and increase the chance of a violation. The separation procedure for the SFCI is derived in a similar way.

Once a violated SFPI is obtained, we lift the inequality to obtain an FPI. First, we set $C_1 = C'_1$, $C_2 = C'_2$ and $\mu = \mu'$. Then, we initialize $D_1 = \{(r, t)\}$ and add to D_1 each arc $(i, j) \in (S, \bar{S}) \backslash C_1$ such that $\bar{x}^L_{ij} - \min\{b^L_{ij}, -\mu\}\bar{y}_{ij} > 0$. Finally, we lift the resulting FPI by applying the function proposed by Atamtürk [5]: we lift all variables in C_1 and the variables in $(\bar{S}, S) \backslash C_2$ such that $\bar{y}_{ij} = 0$. In addition, if $\mu' + b^L_{rt} > 0$, we lift the violated SFPI to generate a violated FCI. We first add (r, t) to C'_1 to obtain C_1, set $C_2 = C'_2$ and compute $\mu = \mu' + b^L_{rt}$. Then, for each arc $(i, j) \in C_1$ such that $b^L_{ij} > \mu$, we add to the left hand side of the inequality the term $(b^L_{ij} - \mu)(1 - y_{ij})$. We then set $D_2 = \{(j, i) \in (\bar{S}, S) \backslash C_2 \mid \bar{x}^L_{ji} > \min\{b^L_{ji}, \mu\}\bar{y}_{ji}\}$. Finally, we lift the resulting FCI by applying the function proposed by Atamtürk [5]: we lift all variables in C_2 and the variables in $(S, \bar{S}) \backslash C_1$ such that $\bar{y}_{ij} = 0$.

We proceed similarly when a violated SFCI is generated. First, we lift the inequality to derive a violated FCI. To this end, we set $C_1 = C'_1$, $C_2 = C'_2$ and $\mu = \mu'$, and then proceed as above to obtain a lifted FCI. If $\mu' - b^L_{rt} < 0$, we also lift the violated SFCI to generate a violated FPI, by setting $C_1 = C'_1$, adding (r, t) to C'_2 to obtain C_2 and computing $\mu = \mu' - b^L_{rt}$; then, we proceed as above to generate a lifted FPI.

To summarize, for each cutset identified by the cutting-plane algorithm, the separation procedure first identifies violated SFPI and SFCI. For each of these violated inequalities, lifting is applied to generate a FCI, a FPI, or both. Our approach to generate FCI and FPI contrasts significantly with the standard separation procedure, which uses a relaxation involving only the 0-1 variables, thus allowing to derive FCI and FPI from simple covers [50]. Here, we use a relaxation that involves both the 0-1 and the continuous variables, allowing us to derive FCI and FPI from single-arc structures.

4 Cutting-Plane Algorithm

The cutting-plane algorithm (Algorithm 2) starts by solving the LP relaxation of formulation (1)-(7), the so-called *weak relaxation* of the problem. Subsequently, it alternates between the generation of cuts and the solution of the current LP relaxation (with the addition of all cuts generated so far). The generation of cuts is controlled by parameters that determine whether or not the separation and lifting procedures for each class of VI should be activated. If the generation of any one of the cutset-based inequalities (i.e., LCI, LMCI, FCI, FPI) is activated, the generation of cuts starts by identifying a family of cutsets. For each cutset in this family, the corresponding violated cutset-based inequalities are generated.

The cutting-plane algorithm follows two phases. In Phase I, the family of cutsets

is based on singletons, i.e., for each cutset $(S, \bar{S})$, S is an origin or $\bar{S}$ is a destination for at least one commodity. Phase I iterates over this family of cutsets until no further significant improvement in the lower bound, z, is observed. In Phase II, more complex families of cutsets are generated, using one of the three approaches described in the remainder of this section. At the end of Phase II, if the bound has improved from the first to the second phase, Phase I is launched all over again. To limit the total computational effort, we use three parameters (their setting is discussed in Section 5.4):

- δ, the minimum bound improvement required to continue the procedure (in Phase I, we compute the improvement between two consecutive LPs; in Phase II, we compare the lower bounds at the beginning and at the end of the phase);
- T_{max}, which limits T, the number of calls to Phase II;
- $\mathbf{M}_{max}$, which is an upper bound on the cardinality of the subsets S generated in Phase II.

Three approaches are used to generate families of cutsets in Phase II (Step 18 of the procedure). The first approach, called *Enumeration*, consists in generating **all** possible subsets of N of cardinality $\mathbf{M}$. Clearly, $\mathbf{M}_{max}$ should then be kept at a relatively small value, otherwise the number of cutsets is prohibitively large. In Section 5.4, the *Enumeration* approach is compared with the two other approaches, described in the next two subsections, which generate only **some** of the subsets of N of cardinality $\mathbf{M}$.

4.1 Articulation Sets and Metric Inequalities

The second approach uses the notion of *articulation set*, which is a set $S \subset N$ such that the removal of S disconnects, for at least one commodity k, its origin $O(k)$ from its destination $D(k)$. Note that if $k \in K(S, \bar{S})$, i.e., $O(k) \in S$ and $D(k) \in \bar{S}$, S is by definition an articulation set for k, but there might be other articulation sets such that $O(k) \in \bar{S}$. To identify all articulation sets of cardinality $\mathbf{M}$, we consider every subset S of cardinality $\mathbf{M}$ and solve shortest path problems for every commodity k with all arc lengths equal to 0, except those of the arcs in $(S, \bar{S})$, which are set to 1. If the shortest path length for commodity k is greater than 0, S is identified as an articulation set. In addition, if the shortest path length for commodity k is greater than 1, this means that every path between $O(k)$ and $D(k)$ must cross $(S, \bar{S})$ more than once. Under this condition, the single-cutset inequality

$$\sum_{(i,j)\in(S,\bar{S})} u_{ij} y_{ij} \geq \sum_{k\in K(S,\bar{S})} d^k \tag{21}$$

is dominated by a metric inequality [51], whose general form is:

$$\sum_{(i,j)\in(S,\bar{S})} u_{ij} y_{ij} \geq \sum_{k\in K} \pi^k_{(S,\bar{S})} d^k, \tag{22}$$

Algorithm 2 CuttingPlane

1: Solve the weak relaxation, yielding z (the optimal value) and $\bar{y}$ (the design solution)
2: **if** $\bar{y}$ is integral **then**
3: stop
4: **end if**
5: $z_{last} \leftarrow z$ and $T \leftarrow 0$
6: `Phase I`: Generate cuts, using the family of cutsets based on all singletons
7: **if** some cuts were found **then**
8: Solve the LP relaxation, yielding z and $\bar{y}$
9: **if** $\bar{y}$ is integral or $z - z_{last} \leq \delta$ **then**
10: stop
11: **end if**
12: $z_{last} \leftarrow z$ and go to 6
13: **end if**
14: `Phase II`:
15: **if** $T < T_{max}$ **then**
16: $z_{last} \leftarrow z$ and $T \leftarrow T + 1$
17: **for** $\mathbf{M} = 2$ to $\mathbf{M}_{max}$ **do**
18: Generate a family of cutsets based on subsets of N of cardinality $\mathbf{M}$
19: Generate cuts, using the current family of cutsets
20: **if** some cuts were found **then**
21: Solve the LP relaxation, yielding z and $\bar{y}$
22: **if** $\bar{y}$ is integral **then**
23: stop
24: **end if**
25: **end if**
26: **end for**
27: **if** $z - z_{last} > \delta$ **then**
28: go to 6
29: **end if**
30: **end if**

where $\pi^k_{(S,\bar{S})}$ is the length of the shortest path between $O(k)$ and $D(k)$ with arc lengths equal to 1 in $(S, \bar{S})$ and 0 everywhere else. Indeed, when S is an articulation set only for the commodities in $K(S, \bar{S})$ and every path between $O(k)$ and $D(k)$ crosses $(S, \bar{S})$ only once, for each commodity $k \in K(S, \bar{S})$, then the single-cutset inequality (21) reduces to (22); otherwise, (21) is dominated by (22). The validity of (22) is easy to prove by using LP duality (see [17] for a complete discussion).

In the so-called *Articulation* approach, we thus generate all cutsets $(S, \bar{S})$ where S is an articulation set of cardinality **M**. The articulation sets for cardinality **M** are generated only once, before the first execution of the **for** loop at Step 17, and the corresponding cutsets are stored in memory for subsequent calls to Phase II (this is also how the *Enumeration* approach is implemented). When violations of the cutset-based inequalities are examined, the constant term $d^L_{(S,\bar{S})}$ is replaced by $\sum_{k\in L} \pi^k_{(S,\bar{S})} d^k$, since the shortest path lengths have already been computed.

4.2 Metaheuristics-Based Cutset Generation

In this third approach, the generation of the corresponding families of cutsets is dynamic, as it depends on the current solution to the LP relaxation. In this *Heuristic* approach, new families of cutsets are obtained by partitioning the set of nodes N into L subsets $S_l, l = 1, ..., L$, such that $S_l \cap S_k = \emptyset$, for all $l \neq k$, and $\cup_{l=1,...,L} S_l = N$. Then, each subset $S_l, l = 1, ..., L$, induces two cutsets $(S_l, \bar{S}_l)$ and $(\bar{S}_l, S_l)$, and the corresponding partition of N determines a family of cutsets available for the generation of violated VI.

This approach is inspired by principles derived from metaheuristics. First, it calls upon a *construction* procedure to provide an initial partition of N into subsets of cardinality **M**. Cuts are generated on this initial family of cutsets. Then, a fixed number, I_{max}, of iterations of a *local search* procedure is performed to derive new partitions of N into subsets of cardinality **M**. Each new partition is obtained by simply moving nodes among subsets around a cycle, thus preserving the subset cardinality from the initial partition to the new one. For each partition thus obtained, cuts are generated for the corresponding family of cutsets. To summarize, in the *Heuristic* approach, the family of cutsets generated at Step 18 is the union of the families of cutsets obtained by the construction procedure and the I_{max} calls to the local search procedure.

The initial partition of N into subsets of cardinality **M** is obtained by the construction procedure called **GenerateMultiSet**(M) (Algorithm 3). Since all types of cutset-based inequalities have a higher chance of being violated when the arcs in $(S_l, \bar{S}_l)$ display small fractional values $\bar{y}_{ij}$, the procedure attempts to construct the sets S_l with the objective of minimizing $\sum_{(i,j)\in(S_l,\bar{S}_l)} \bar{y}_{ij}$ and $\sum_{(j,i)\in(\bar{S}_l,S_l)} \bar{y}_{ji}$. At any step of the procedure, let S_l be a subset of N of cardinality smaller than **M**. Initially, the family contains one subset, S_1, having a single element (arbitrarily chosen). We denote *free node*, a node that is not included in any subset, and $\bar{N}$, the set of all free nodes. Also, for each free node j, let

$$w_j = \max\{\max_{i\in S_l} \bar{y}_{ij}, \max_{i\in S_l} \bar{y}_{ji}\}.$$

To achieve our objective, we identify the free node n such that $n = \text{argmax}_{j\in\bar{N}} \{w_j\}$. If n

exists, then we add it to S_l and move to the next step: continue with the construction of S_l, if $|S_l| < \mathbf{M}$ or, otherwise, proceed to the construction of S_{l+1} (by selecting arbitrarily some free node and then repeating the process). If, however, no free node is connected by an arc to at least one node in S_l, we choose n arbitrarily among the free nodes. The procedure stops when there are no more free nodes.

Algorithm 3 GenerateMultiSet(M)

1: $\bar{N} \leftarrow N$, $l \leftarrow 1$
2: **if** $\bar{N} = \emptyset$ **then**
3: stop
4: **end if**
5: Select (arbitrarily) a node $m \in \bar{N}$
6: Add m to S_l and remove it from $\bar{N}$
7: **if** $|S_l| \geq \mathbf{M}$ **then**
8: $l \leftarrow l + 1$ and go to 2
9: **end if**
10: $n \leftarrow \text{argmax}_{j \in \bar{N}} \{w_j\}$
11: **if** n exists **then**
12: $m \leftarrow n$ and go to 7
13: **end if**
14: Go to 2

Note that the procedure attempts to first include in S_l a free node that is connected by an arc to at least one node in S_l to avoid generating VI that are aggregations of previously generated VI. Indeed, sets S_l must be connected, otherwise the corresponding cutset-based inequalities will be dominated by others. Our construction procedure is similar to the heuristic methods used in [11, 28, 52, 54], in that these approaches also build a subset S by starting from a single node and by gradually enlarging it through the addition of neighboring nodes that are connected by an arc to at least one node in S. The difference lies in the criteria being used to add a neighboring node; [52] use the same criterion as ours, but also other criteria, while [11, 28, 54] use the sum of the slack and the value of the dual variable in the capacity constraint. The local search procedure, that we now present, has no analog in the literature, to the best of our knowledge.

The local search procedure identifies new families of cutsets by performing exchanges of nodes among subsets of the current family. The basic idea behind these exchanges is to obtain a new subset $S_{l'}$ from a subset S_l by moving a node n from some set S_k, $S_k \subset \bar{S}_l$, to S_l. These exchanges are performed by the procedure **MultiExchange**$((S_l)_{l=1,\ldots,L}, W, W_N)$, illustrated in Algorithm 4.The sets W and W_N contain, respectively, the indices l of all subsets S_l and the nodes $n \in N$ involved in some exchanges at previous calls to the procedure. These sets are used to ensure that the exchanges reach different subsets and involve different nodes, thus creating new cutsets at each iteration. The procedure considers at each step a set S_l and aims to identify and move to S_l the node n such that

$$n = \text{argmax}_{j \in (N \setminus W_N) \cap (\cup_{k \notin W, S_k \subset \bar{S}_l} S_k)} \{w_j\}.$$

Note that $n \in N \backslash W_N$ is chosen among the set of nodes connected by an arc to at least one node in S_l. Again, this strategy attempts to avoid generating VI that are aggregations of previously generated ones. Once n is identified, we move it from some set S_k to S_l. Then, the procedure repeats the process by considering subset S_k at the next iteration. The procedure starts with a set S_l not involved in previous exchanges (i.e., $l \notin W$). The procedure also stores in set V the indices of the subsets S_l considered at each iteration and stops whenever it finds a couple of subsets (S_l, S_k) involved in an exchange such that $k \in V$. This strategy identifies a cycle on which the nodes are moved around. By doing so, all subsets have the same cardinality as before the exchanges.

Algorithm 4 MultiExchange$((S_l)_{l=1,...,L}, W, W_N)$

1: $V \leftarrow \emptyset$
2: **if** $W = \{1, ..., L\}$ **then**
3: $\quad W \leftarrow \emptyset$
4: **end if**
5: Let $l \notin W$ correspond to some set not involved in previous exchanges
6: $n \leftarrow \text{argmax}_{j \in (N \backslash W_N) \cap (\cup_{k \notin W, S_k \subset \bar{S}_l} S_k)} \{w_j\}$
7: **if** $(\cup_{k \notin W, S_k \subset \bar{S}_l} S_k) = \emptyset$ **then**
8: $\quad W \leftarrow \emptyset$ and go to 5
9: **end if**
10: **if** n does not exist **then**
11: $\quad W_N \leftarrow \emptyset$ and go to 5
12: **end if**
13: Let $S_k \subset \bar{S}_l$ such that $n \in S_k$
14: Move n from S_k to S_l
15: $W_N \leftarrow W_N \cup \{n\}$
16: $W \leftarrow W \cup \{l\}$
17: **if** $V = \emptyset$ **then**
18: $\quad l_0 \leftarrow l$
19: **end if**
20: $V \leftarrow V \cup \{l\}$
21: **if** $k \in V$ **then**
22: $\quad$ **if** $k \neq l_0$ **then**
23: $\quad\quad n \leftarrow \text{argmax}_{i \in S_{l_0}} (\max_{j \in S_k} \bar{y}_{ij}, \max_{j \in S_k} \bar{y}_{ji})$
24: $\quad\quad$ Move n from S_{l_0} to S_k (to complete the cycle)
25: $\quad$ **end if**
26: $\quad$ Stop
27: **end if**
28: $l \leftarrow k$ and go to 6

5 Computational Results

Computational experiments were performed with four objectives in mind: 1) Verify that our implementation of separation and lifting procedures for CI and FCI is competitive with that of a state-of-the-art MIP solver (CPLEX, version 12); 2) Compare the relative performance of the different classes of VI; 3) Test the performance of the cutset generation procedures; 4) Evaluate the quality of the formulations obtained from different variants of the cutting-plane algorithm, by performing the B&B algorithm of CPLEX (version 12) on each of these formulations. Following a first section that describes the data instances and the performance measures used in the experiments, we present and analyze the results in the four subsequent subsections, each dedicated to one of these objectives. To facilitate reading, we recall the abbreviations used for each class of inequalities: SI, strong inequalities (9); CI, cover inequalities (13); MCI, minimum cardinality inequalities (14); FCI, flow cover inequalities (17); FPI, flow pack inequalities (18).

5.1 Data Instances and Performance Measures

Computational experiments were conducted on a publicly available set of 196 instances (the so-called "Canad" instances [26]) used in several papers on the $MCND$ (for instance [31, 37, 40]) and described in detail by Crainic *et al.* [21]. These problem instances consist of general transshipment networks with one commodity per origin-destination and no parallel arcs. Associated with each arc are three positive quantities: the capacity, the fixed charge, and the transportation cost. These instances are characterized by various degrees of capacity tightness, with regard to the total demand, and importance of fixed design cost, with respect to the transportation cost.

The instances are divided into three classes. Class I (the "C" instances in [26]) consists of 31 problem instances with many commodities compared to the number of nodes, while Class II (the "C+" instances in [26]) contains 12 problem instances with few commodities compared to the number of nodes. Class III (the "R" instances in [26]) is divided into two categories, A and B, each containing nine sets of nine problem instances each. Each set is characterized by the numbers of nodes, arcs, and commodities, which are the same for the nine instances, and by instance-specific levels of fixed cost and capacity tightness. Class III-A (instances "R01" to "R09") contains 72 small size problem instances with 10 nodes (nine infeasible instances have been discarded), while Class III-B (instances "R10" to "R18") contains 81 medium to large size instances with 20 nodes.

To evaluate the performance of the different formulations and variants of the cutting-plane algorithm, we use three measures:

- The computing time, t, where all experiments are performed on a network of Dual-Core AMD Opteron (using a single thread) with 8 Gigabytes of RAM operating under SunOS 5.10. The procedures are coded in C++. To solve the LP relaxations, we use the dual simplex implementation of CPLEX, version 12.

- The gap between the lower bound and the value of a reference solution. For the weak relaxation, we use as reference solution the best (often optimal) solution of

Description	Nb	Weak LP Δz^*	t(Dis)	t(Agg)	Description	Nb	Weak LP Δz^*	t(Dis)	t(Agg)
Class I					Class II				
20,230,40	(3)	7.70%	0.1	0.1	25,100,10	(3)	29.02%	0.0	0.0
20,230,200	(4)	26.50%	1.1	0.9	25,100,30	(3)	24.44%	0.1	0.1
20,300,40	(4)	9.74%	0.1	0.1	100,400,10	(3)	37.25%	0.6	0.9
20,300,200	(4)	21.01%	1.6	1.1	100,400,30	(3)	34.01%	1.7	2.6
30,520,100	(4)	19.51%	0.8	0.9					
30,520,400	(4)	15.32%	6.5	7.9					
30,700,100	(4)	17.72%	0.7	0.9					
30,700,400	(4)	17.67%	7.8	7.3					
Average	(31)	17.19%	2.4	2.5	Average	(12)	31.18%	0.6	0.9
Class III-A					Class III-B				
10,35,10	(6)	12.61%	0.0	0.0	20,120,40	(9)	21.93%	0.1	0.1
10,35,25	(6)	17.96%	0.0	0.0	20,120,100	(9)	19.56%	0.6	0.5
10,35,50	(6)	14.34%	0.0	0.0	20,120,200	(9)	16.68%	2.3	0.8
10,60,10	(9)	20.26%	0.0	0.0	20,220,40	(9)	29.91%	0.2	0.2
10,60,25	(9)	16.06%	0.0	0.0	20,220,100	(9)	26.84%	0.6	0.6
10,60,50	(9)	18.67%	0.0	0.0	20,220,200	(9)	23.87%	4.4	1.1
10,85,10	(9)	17.25%	0.0	0.0	20,320,40	(9)	32.30%	0.2	0.2
10,85,25	(9)	18.69%	0.0	0.0	20,320,100	(9)	30.27%	0.7	0.8
10,85,50	(9)	21.54%	0.0	0.0	20,320,200	(9)	27.70%	5.1	1.4
Average	(72)	17.80%	0.0	0.0	Average	(81)	25.45%	1.6	0.6

Table 1: Classes and Problem Dimensions

value z^* obtained by using the B&B algorithm of CPLEX (version 12) for a limit of 10 hours on several formulations derived from the cutting-plane algorithm (see Section 5.5 for a description of these formulations). For the weak relaxation lower bound z^w, we thus report the following gap measure:

$$\Delta z^* = \frac{100(z^* - z^w)}{z^*}.$$

For any lower bound z computed by the cutting-plane procedure, the reference is the weak relaxation bound, and we use the following gap measure:

$$\Delta z^w = \frac{100(z - z^w)}{z^w}.$$

- The number of cuts generated by the cutting-plane algorithm.

Table 1 gives the classification of the instances. Columns "*Description*" and "*Nb*" show the dimension of the instances, characterized by the numbers of nodes, arcs, and commodities, and the number of instances with these dimensions, respectively. The average gap between the bounds of the weak relaxation and the best known feasible solution is given under column Δz^*, while the average times required to solve the weak relaxation for the disaggregated and aggregated formulations are given in columns t(Dis) and t(Agg), respectively. The "*Average*" line shows the gap average over all instances in each class along with the average times required to compute the bounds. The results in column Δz^* confirm the poor quality of the lower bounds generated by the weak relaxation. Obviously, the disaggregated and aggregated formulations provide the same lower bound, and they do so with a similar (and negligible) computational effort.

5.2 Comparison with CPLEX Cuts

Table 2 displays the per-class average results obtained by the cutting-plane method implemented in CPLEX (version 12), with default settings, and those of our cutting-plane algorithm, using the disaggregated and aggregated commodity representations. We aim especially to compare the respective implementations of CI and FCI. For a fair comparison, single-node cutset structures have been added to the formulations given to CPLEX. These special structures are redundant in the formulation but allow CPLEX to identify violated cover inequalities. The columns "*CI*" and "*FCI*" display, respectively, the average results obtained by using CI alone and FCI alone, while the columns "*All*" and "*Enum1*" show the average results obtained by using all classes of VI in "*CPLEX*" and our "*Cutting-Plane*" algorithm. Note that "*Enum1*" denotes the variant of our cutting-plane algorithm that performs only Phase I, i.e., all classes of VI are used, but only single-node cutsets are used in the cutset generation procedure.

The results indicate that our implementation of separation and lifting procedures for CI and FCI is competitive with that of a general-purpose state-of-the-art MIP solver. Indeed, we observe, when generating cover inequalities, better gap improvements with our implementation on the disaggregated models and the opposite on the aggregated models; in all cases, the differences both in terms of gap improvement, time and number of cuts are relatively minor. When generating flow cover inequalities, our implementation provides better gap improvements on average, in much less time for the disaggregated models (even though about three times more cuts on average are generated by our implementation) and in slightly more time for the aggregated models. This shows that our separation method for FCI provides effective results for our $MCND$ instances; it would be interesting to evaluate the performance of this separation method on general MIPs.

We have added the results with all classes of cuts implemented in CPLEX and in our algorithm to show that, even in that case, our results remain competitive. For the disaggregated models, the gap improvements are better on average, but these improvements are obtained with much less computational effort; for the aggregated models, the gap improvements are also better on average, but the computing times are slightly higher. We note, however, that the number of cuts generated by our implementation is significantly larger than the number of cuts generated by CPLEX. This is certainly a concern when implementing our cutting-plane method within a B&B framework, but only moderately so, since procedures to remove inactive cuts can be easily added to ensure the size of the formulations remains tractable.

These results also suggest the following observations, which we will confirm with further experiments:

- Based on the gap improvements, we can conclude that cover inequalities (with Δz^w always less than 10% on average) are dominated by flow cover inequalities (with Δz^w always larger than 20% on average). This is true not only for our algorithm, but also for CPLEX. Given the fact that CI is derived from the single-cutset relaxation, which is itself a relaxation of the single-cutset flow structure, from which we obtain FCI, the dominance of FCI over CI was expected, but not the extent by which FCI dominates CI.

- Flow cover inequalities capture most of the lower bound improvement coming from all types of cuts; this is true for our cutting-plane algorithm, but also for CPLEX. These results confirm the literature on fixed-charge network design that identifies FCI as strong VI for such problems.
- The disaggregated commodity representation provides better lower bound improvements than the aggregated one, at the expense of higher computing times; again, this is true for our cutting-plane algorithm and for CPLEX. The two algorithms differ significantly, however, in their respective computational effort to handle the disaggregated commodity representation, our implementation being an order of magnitude faster on average, in spite of generating a significantly larger number of cuts.

CPLEX (Disaggregated)

Classes		CI			FCI			All		
	Nb	Δz^w	t	Cuts	Δz^w	t	Cuts	Δz^w	t	Cuts
Class I	(31)	0.48%	10.1	7	17.95%	1606.8	662	18.75%	993.1	696
Class II	(12)	23.26%	0.1	18	38.19%	0.8	110	47.53%	4.7	151
Class III-A	(72)	5.39%	0.0	5	17.71%	0.1	72	21.00%	0.1	82
Class III-B	(81)	4.59%	2.4	13	27.87%	101.8	333	31.23%	104.1	384
Average	(196)	5.37%	2.6	9	23.20%	296.3	276	26.49%	200.4	308

Cutting-Plane (Disaggregated)

Classes		CI			FCI			Enum1		
	Nb	Δz^w	t	Cuts	Δz^w	t	Cuts	Δz^w	t	Cuts
Class I	(31)	1.00%	2.5	21	17.32%	158.2	1288	19.21%	95.5	3123
Class II	(12)	24.38%	0.7	28	50.66%	10.1	725	52.70%	5.3	1402
Class III-A	(72)	8.78%	0.0	16	20.35%	0.1	165	21.25%	0.1	363
Class III-B	(81)	8.86%	2.9	38	31.34%	63.4	1125	33.71%	21.5	2343
Average	(196)	8.54%	1.6	27	26.27%	51.9	774	28.00%	24.3	1682

CPLEX (Aggregated)

Classes		CI			FCI			All		
	Nb	Δz^w	t	Cuts	Δz^w	t	Cuts	Δz^w	t	Cuts
Class I	(31)	0.12%	1.3	7	13.83%	26.6	584	16.28%	28.8	463
Class II	(12)	16.57%	0.1	18	38.24%	0.6	109	46.12%	4.3	147
Class III-A	(72)	5.63%	0.0	6	16.89%	0.0	57	19.93%	0.0	68
Class III-B	(81)	2.68%	0.2	14	24.02%	2.8	238	25.92%	6.4	264
Average	(196)	4.21%	0.3	10	20.66%	5.4	218	23.43%	7.5	216

Cutting-Plane (Aggregated)

Classes		CI			FCI			Enum1		
	Nb	Δz^w	t	Cuts	Δz^w	t	Cuts	Δz^w	t	Cuts
Class I	(31)	0.12%	2.2	2	13.30%	43.0	2572	14.09%	51.9	4825
Class II	(12)	9.80%	0.9	7	49.31%	11.7	754	51.40%	7.6	1469
Class III-A	(72)	2.33%	0.0	2	18.55%	0.1	184	19.36%	0.1	363
Class III-B	(81)	1.85%	0.6	4	26.32%	11.3	1240	27.87%	19.5	2486
Average	(196)	2.24%	0.7	3	22.81%	12.2	1033	24.01%	16.8	2014

Table 2: CPLEX Cuts Versus Cutting-Plane Algorithm

5.3 Comparison Among Classes of Valid Inequalities

In this section, we present the results of computational experiments performed to compare the relative performance of the five classes of VI. As in the previous section, only Phase

I of the cutting-plane algorithm was performed. We first present average results over all classes of instances and then, we analyze the results for the different classes of instances.

Table 3 shows, for the disaggregated and aggregated commodity representations, the improvement gap, Δz^w, and the computing time, t, averaged over the 196 instances. In column "*None+*" we show the results obtained by using each individual class alone, while in column "*All-*" we display the results obtained by using all classes of VI, except the one identifying the respective row. These results show the superiority of the inequalities based on continuous and 0-1 variables, i.e., SI, FCI, and FPI, over those based only on 0-1 variables, i.e., CI and MCI. They also show that the disaggregated commodity representation provides tighter formulations than the aggregated commodity representation. In particular, by adding only the SI to the disaggregated model, we obtain better lower bounds on average than by adding all the cuts to the aggregated model, in about the same computational effort. It is interesting to note that, for both commodity representations, the FPI is the most effective individual class of inequalities for improving the bound, but at the expense of a significant computational effort. In particular, the SI class is almost as effective as the FPI class for the disaggregated representation, but adding SI requires much less computing time. In fact, the SI are essential for obtaining good performance: removing them leads to significant increase in computing time.

	Disaggregated				Aggregated			
	None+		All-		None+		All-	
	Δz^w	t	Δz^w	t	Δz^w	t	Δz^w	t
$\emptyset$	0%	1.1	28.00%	24.3	0%	0.7	24.01%	16.7
SI	26.53%	17.8	27.12%	48.1	11.73%	0.8	24.01%	21.5
CI	8.54%	1.6	27.96%	25.3	2.24%	0.7	24.02%	15.8
MCI	8.00%	1.5	28.00%	24.4	2.09%	0.7	24.03%	14.9
FCI	26.27%	51.9	28.01%	26.5	22.81%	12.2	23.99%	11.3
FPI	26.89%	51.2	27.94%	21.0	23.95%	17.6	22.87%	9.3

Table 3: Comparison of Valid Inequalities for All Instances

Table 4 analyzes, for each class of instances, the effect of activating each individual class of inequalities, for the disaggregated and aggregated commodity representations. Results are reported only for the inequalities that involve continuous and 0-1 variables, i.e., SI, FPI, and FCI, which have already been shown to be stronger than the other classes of inequalities.

These results emphasize the differences between the aggregated and disaggregated commodity representations. Not only the SI is significantly more effective in improving the lower bound within the disaggregated representation, as expected, but also the FCI and FPI reduce the lower bound gap more significantly within the disaggregated representation, and by generating less cuts. We note that the differences between the two commodity representations are less pronounced for Class II and, to a certain extent, for Class III-A. This is not surprising, as instances in Classes II and III-A are characterized by a small number of commodities, and disaggregation has less impact on such instances. With the disaggregated commodity representation, SI shows the best overall performance regarding the lower bound improvement and the computational effort needed. For instances in Classes II and III-A, however, FPI obtains the best average gap improvement, but at the expense of increasing the computing time. When the number of commodities

Disaggregated										
Classes		SI			FCI			FPI		
	Nb	Δz^w	t	Cuts	Δz^w	t	Cuts	Δz^w	t	Cuts
Class I	(31)	19.06%	80.1	870	17.32%	158.2	1288	17.89%	139.1	1941
Class II	(12)	44.83%	2.2	208	50.66%	10.1	725	52.15%	14.9	852
Class III-A	(72)	19.52%	0.0	76	20.35%	0.1	165	20.75%	0.1	227
Class III-B	(81)	32.92%	12.1	685	31.34%	63.4	1125	32.06%	68.3	1692
Average	(196)	26.53%	17.8	462	26.27%	51.9	774	26.89%	51.2	1142

Aggregated										
Classes		SI			FCI			FPI		
	Nb	Δz^w	t	Cuts	Δz^w	t	Cuts	Δz^w	t	Cuts
Class I	(31)	2.22%	2.3	58	13.30%	43.0	2572	14.05%	40.9	2925
Class II	(12)	41.18%	2.7	179	49.31%	11.7	754	51.08%	10.4	857
Class III-A	(72)	9.04%	0.0	24	18.55%	0.0	184	19.30%	0.1	222
Class III-B	(81)	13.39%	0.7	119	26.32%	11.3	1240	27.84%	25.4	1706
Average	(196)	11.73%	0.8	78	22.81%	12.2	1033	23.95%	17.6	1302

Table 4: Comparison of Valid Inequalities by Classes of Instances

is significantly larger than the number of nodes, as for most instances in Classes I and III-B, SI outperforms FPI and FCI. Not only the identification of violated VI is easier with SI, but also the number of cuts generated by SI is significantly less than with FCI and FPI.

5.4 Evaluation of Cutset Generation Procedures

In this section, we assess the cutset generation approaches presented in Section 4. More precisely, the following variants of the cutting-plane algorithm were implemented and tested (all classes of VI were used):

- *Enum1*: This variant consists in performing only Phase I, i.e., only single-node cutset structures are considered.

- *Enumj*, $j \geq 2$: These variants are obtained by using the *Enumeration* approach in Phase II, i.e., all subsets of N of cardinality j are generated. We report the results for two values of j: 2 and 3. As we will see below, the bound improvement from *Enum*2 to *Enum*3 is minor, in spite of a significantly increased computational effort.

- *Artic*: This is the *Articulation* approach with $\mathbf{M}_{max} = 2$, i.e., we generate all cutsets $(S, \bar{S})$, where S is an articulation set of cardinality 2.

- *Heur*: This is the *Heuristic* approach based on the construction and local search procedures, **GenerateMultiSet** and **MultiExchange**, presented in Section 4.2. The parameters of the procedures were calibrated and the following values were used: $\delta = 0.1$, $T_{max} = 5$, $\mathbf{M}_{max} = \left\lceil \frac{N}{3} \right\rceil$, and $I_{max} = 20$.

- *ArticHeur*: This variant combines the last two methods. More specifically, articulation sets of cardinality 2 are stored in memory, and when Phase II is launched to

generate cutsets corresponding to subsets of cardinality 2, these articulation sets are first considered before the *Heuristic* approach is performed.

Disaggregated										
Classes		Enum1			Enum2			Enum3		
	Nb	Δz^w	t	Cuts	Δz^w	t	Cuts	Δz^w	t	Cuts
Class I	(31)	19.21%	95.5	3123	19.22%	94.7	3137	19.22%	130.8	3152
Class II	(12)	52.70%	5.3	1402	54.65%	22.9	1684	55.41%	219.2	2304
Class III-A	(72)	21.25%	0.1	363	21.53%	0.1	389	21.64%	0.2	410
Class III-B	(81)	33.71%	21.5	2343	33.78%	22.9	2375	33.81%	25.9	2407
Average	(196)	28.00%	24.3	1682	28.26%	25.9	1723	28.36%	44.9	1785
Classes		Artic			Heur			ArticHeur		
	Nb	Δz^w	t	Cuts	Δz^w	t	Cuts	Δz^w	t	Cuts
Class I	(31)	19.22%	107.8	3138	19.22%	106.5	3137	19.23%	101.5	3149
Class II	(12)	54.36%	16.9	1580	55.15%	18.1	1735	55.36%	24.7	1769
Class III-A	(72)	21.53%	0.1	386	21.52%	0.2	382	21.56%	0.2	389
Class III-B	(81)	33.78%	21.0	2373	33.81%	26.6	2376	33.81%	22.6	2384
Average	(196)	28.24%	26.8	1716	28.29%	29.0	1725	28.32%	27.0	1734
Aggregated										
Classes		Enum1			Enum2			Enum3		
	Nb	Δz^w	t	Cuts	Δz^w	t	Cuts	Δz^w	t	Cuts
Class I	(31)	14.09%	51.86	4825	15.27%	156.1	5514	15.96%	555.5	6418
Class II	(12)	51.40%	7.63	1469	53.95%	33.9	1737	55.13%	634.4	2448
Class III-A	(72)	19.36%	0.1	363	20.65%	0.2	438	21.08%	0.9	511
Class III-B	(81)	27.87%	19.5	2486	29.15%	45.1	2857	29.86%	125.9	3291
Average	(196)	24.01%	16.7	2014	25.35%	45.5	2320	25.98%	179.0	2713
Classes		Artic			Heur			ArticHeur		
	Nb	Δz^w	t	Cuts	Δz^w	t	Cuts	Δz^w	t	Cuts
Class I	(31)	15.19%	149.0	5462	15.40%	165.2	5423	15.71%	208.3	5672
Class II	(12)	53.63%	24.3	1682	54.69%	32.1	1860	54.72%	38.6	1863
Class III-A	(72)	20.50%	0.2	433	20.87%	0.3	430	21.00%	0.4	464
Class III-B	(81)	29.02%	39.9	2827	29.48%	55.0	2830	29.70%	64.2	2958
Average	(196)	25.21%	41.6	2294	25.63%	50.9	2299	25.82%	62.0	2404

Table 5: Evaluation of Cutset Generation Procedures

Even though the disaggregated representation was shown superior to the aggregated one in the previous section, this was only for single-node cutset structures. The situation might change if we allow cutsets based on node subsets of higher cardinality; hence, we report results for the two commodity representations. Table 5 displays the average results obtained by the different cutset generation methods. Methods *Enum2* and *Enum3* are used as a basis of comparison for the *Articulation* and *Heuristic* approaches. Since *Artic* builds a partial list of cutsets based on node subsets of cardinality 2, its lower bound should be less than the one obtained with *Enum2*, which performs complete enumeration of subsets of cardinality 2. As we can see for both commodity representations, the lower bounds obtained by *Artic* are very close to those computed with *Enum2*, with a comparable computational effort. Concerning the performance of the *Heuristic* approach, we note that this method generates cutsets based on node subsets of cardinality 2 or more; hence, *Enum3* can be used as a basis of comparison for *Heur* and *ArticHeur*. We can see that these two methods obtain lower bounds that are extremely close to those generated by *Enum3* and in less computing time (sometimes significantly so, see for instance the results for Class II instances, which have more nodes than the other instances). These

results demonstrate the effectiveness, as well as the computational efficiency, of the *Articulation* and *Heuristic* approaches for generating cutset-based inequalities from node subsets of cardinality 2 or more. In spite of this, including such inequalities provides very little bound improvement on average: less than 2% for the aggregated models and as little as 0.4% for the disaggregated ones. We note, however, that some instances in Class II show more significant bound improvement. These results also confirm the superiority of the disaggregated commodity representation: the lower bounds are not only better, but they are also obtained in much less computing time and by generating less cuts.

5.5 Evaluation of Cutting-Plane Formulations

To evaluate more precisely the models obtained by different variants of our cutting-plane algorithm, we perform the B&B algorithm of CPLEX (version 12) with default options, except for a time limit of 2 hours (note that CPLEX will then generate its own cuts, according to the default options). For each instance, the best known feasible solution (which is often optimal) is provided as the initial incumbent. This way, our experiments focus only on the quality of the lower bounds in terms of their ability to prune the search tree. We perform experiments with the disaggregated and aggregated commodity representations for the following formulations:

- *Weak:* Model (1)-(7).
- *Strong:* Model (1)-(7) plus the SI identified by our cutting-plane algorithm.
- *Enum1:* Model (1)-(7) plus all the VI identified by our cutting-plane algorithm, using SI and cutset-based inequalities derived only from single-node subsets.
- *ArticHeur:* Model (1)-(7) plus all the VI identified by our cutting-plane algorithm, using SI and cutset-based inequalities derived from the *ArticHeur* method.

Following the experiments with the resulting eight formulations, we classify the instances into three classes:

- *Easy:* Instances that are solved to optimality by CPLEX for the eight models.
- *Difficult:* Instances that cannot be solved by CPLEX (within the time limit of 2 hours) for any of the eight formulations.
- *Medium:* Instances that are solved by CPLEX for at least one of the eight formulations.

The results for the 123 *Easy* instances are provided in Table 6, which gives the number of instances in each class, "*Nb*", and for each tested model, the average number of nodes generated in the B&B tree, "*Nodes*", and the average computing time, "*t*". These results show that weak and strong aggregated models perform better for solving the easy instances in Classes III-A and III-B, with very close results for the disaggregated strong formulation, which generates less B&B nodes, but in slightly more computing time. For

the easy instances in Class II, the aggregated model obtained from "Enum1" performs best, with the disaggregated strong formulation a solid second in terms of computing time (although the number of nodes is significantly larger). For Class I instances, the disaggregated models perform much better than the aggregated ones, the disaggregated formulation obtained from "Enum1" performing slightly better than the strong model.

Disaggregated									
Classes		Weak		Strong		Enum1		ArticHeur	
	Nb	Nodes	t	Nodes	t	Nodes	t	Nodes	t
Class I	(7)	355	6.1	231	2.7	181	2.6	223	4.1
Class II	(9)	14667	417.8	15405	349.1	8070	369.1	5025	691.9
Class III-A	(72)	238	3.5	248	2.9	202	4.3	189	5.1
Class III-B	(35)	2174	446.7	2087	264.6	1500	329.7	1361	323.3
Aggregated									
Classes		Weak		Strong		Enum1		ArticHeur	
	Nb	Nodes	t	Nodes	t	Nodes	t	Nodes	t
Class I	(7)	914	33.7	909	33.1	1228	63.6	448	24.9
Class II	(9)	17863	390.7	24300	509.7	6926	345.4	5615	353.4
Class III-A	(72)	346	2.0	372	2.2	361	4.7	325	5.25
Class III-B	(35)	4147	219.9	3935	214.8	4836	674.6	5335	714.8

Table 6: CPLEX B&B, 2 hours CPU Time Limit, Easy Instances

The results for the 52 *Difficult* instances are provided in Table 7, which gives the number of instances in each class, "*Nb*", and for each tested model, the average number of nodes generated in the B&B tree, "*Nodes*", and the average final gap between the best lower and upper bounds, "Δz^*". These results show the superiority of the disaggregated commodity representation for difficult instances. Indeed, the final gap is generally smaller after 2 hours of computing time, even though the number of generated nodes is significantly smaller with the disaggregated models, which can be explained by the larger size of the disaggregated LP relaxations solved at every node of the B&B tree. The disaggregated strong model gives the best results for the difficult instances in Classes I and III-B, which are characterized by few nodes (less than 30) and many commodities (more than 100). The "ArticHeur" and "Enum1" variants perform best for the difficult instances in Class II, which have many nodes (100) and few commodities (30). Even for these instances, the final gap obtained by the disaggregated strong formulation is close to the best final gap computed with the "ArticHeur" and "Enum1" approaches.

The results for the 21 *Medium* instances are provided in Table 8, which gives the number of instances in each class, "*Nb*", and for each tested model, the number of instances solved by CPLEX within the time limit of 2 hours, "*Sol*", the average time necessary to solve these instances, "t(Sol)", and the average final gap for the instances that could not be solved by CPLEX within 2 hours, "Δz^*". The results show the superiority of the disaggregated strong model for solving these instances: the three instances in Class I are solved (with "Enum1" also, but in more computing time), while 13 of the 18 instances in Class III-B are solved to optimality. The disaggregated weak and the aggregated strong formulations also solve 13 instances in Class III-B, but in more computing time. In addition, the final gap for the remaining unsolved instances is smaller for the disaggregated strong model.

Disaggregated									
Classes		Weak		Strong		Enum1		ArticHeur	
	Nb	Nodes	Δz^*	Nodes	Δz^*	Nodes	Δz^*	Nodes	Δz^*
Class I	(21)	3741	1.66%	4563	1.54%	3332	1.56%	2511	1.56%
Class II	(3)	25121	5.40%	23108	4.35%	7062	3.98%	2462	3.92%
Class III-B	(28)	5930	2.35%	6924	2.22%	3094	2.30%	2536	2.41%
Aggregated									
Classes		Weak		Strong		Enum1		ArticHeur	
	Nb	Nodes	Δz^*	Nodes	Δz^*	Nodes	Δz^*	Nodes	Δz^*
Class I	(21)	9172	2.63%	10143	2.59%	2973	3.41%	2458	3.31%
Class II	(3)	27207	6.14%	23748	4.67%	5804	4.17%	3067	4.20%
Class III-B	(28)	25256	3.62%	25125	3.44%	12562	4.26%	10579	4.19%

Table 7: CPLEX B&B, 2 hours CPU Time Limit, Difficult Instances

Disaggregated													
Classes		Weak			Strong			Enum1			ArticHeur		
	Nb	Sol	t(Sol)	Δz^*	Sol	t(Sol)	Δz^*	Sol	t(Sol)	Δz^*	Sol	t(Sol)	Δz^*
Class I	(3)	2	2354.5	0.08%	3	2361.3	-	3	3934.3	-	1	60	0.07%
Class III-B	(18)	13	2308.5	0.74%	13	1871.8	0.58%	10	2841.3	0.67%	9	3067.4	0.66%
Aggregated													
Classes		Weak			Strong			Enum1			ArticHeur		
	Nb	Sol	t(Sol)	Δz^*	Sol	t(Sol)	Δz^*	Sol	t(Sol)	Δz^*	Sol	t(Sol)	Δz^*
Class I	(3)	0	-	0.70%	1	6913.0	0.84%	0	-	1.56%	0	-	1.33%
Class III-B	(18)	11	2391.5	1.01%	13	3198.3	0.71%	4	3930.5	1.89%	2	1206.0	1.56%

Table 8: CPLEX B&B, 2 hours CPU Time Limit, Medium Instances

Over all classes of instances, the disaggregated strong model emerges as the most effective one. It solves 139 instances in the smallest average computing time among the eight modeling options, while the remaining 57 unsolved instances display an average optimality gap of 1.93%. In general, adding cutset-based inequalities yields LP relaxations that are too large, which generally translates into prohibitive computational effort, although some instances in Class II can be solved more efficiently with the introduction of flow pack inequalities. The aggregated commodity representation is generally outperformed by the disaggregated one, except for the easiest instances; even for these instances, the disaggregated models perform well.

6 Conclusions

In this paper, we have presented a cutting-plane algorithm for the multicommodity capacitated fixed-charge network design problem. We have described five families of known VI: the strong, cover, minimum cardinality, flow cover, and flow pack inequalities. We have developed efficient separation and lifting procedures, as well as a cutset generation algorithm based on metaheuristics principles. Finally, we have presented computational results conducted on a large set of instances. Our computational experiments have focused on two key modeling aspects: the representation of the commodities, either aggregated or disaggregated, and the impact of the cutset-based inequalities.

Our computational study shows the strength of the disaggregated commodity representation, when combined with dynamic generation of strong inequalities. It also suggests that cutset-based inequalities have a limited impact on instances with many commodities (more than 100). Although we have tested our cutting-plane algorithm within the enumerative framework implemented in CPLEX, the procedure can be included into a more promising custom-made branch-and-cut algorithm. Finally, it would be interesting to investigate the usefulness of the proposed separation and cutset generation methods to improve the formulations of other network design formulations.

Acknowledgments

While working on this project, the first author was post-doctoral fellow with the NSERC Industrial Research Chair on Logistics Management, ESG UQAM. The second author held that Chair and was Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal, and the Department of Economics and Business Administration, Molde University College, Norway. Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grants programs, by the partners of the Chair, CN, Rona, Alimentation Couche-Tard and the Ministry of Transportation of Québec, and by the Fonds québécois de recherche sur la nature et les technologies (FQRNT Québec) through its Team Research grants program. We want to take this opportunity to thank Mrs. Geneviève Hernu, analyst with the Chair, for her very important participation in setting up the codes and performing the experiments.

References

[1] K. Aardal. Capacitated facility location: separation algorithms and computational experience. *Mathematical Programming*, 81:149–175, 1998.

[2] K. Aardal, Y. Pochet, and L.A. Wolsey. Capacitated facility location: valid inequalities and facets. *Mathematics of Operations Research*, 20:562–582, 1995.

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

[4] A. Armacost, C. Barnhart, and K.A. Ware. Composite variable formulations for Express Shipment Service Network Design. *Transportation Science*, 36:1–20, 2002.

[5] A. Atamtürk. Flow pack facets of the single node fixed-charge flow polytope. *Operations Research Letters*, 29:107–114, 2001.

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

[7] A. Atamtürk and O. Günlük. Network design arc set with variable upper bounds. *Networks*, 50:17–28, 2007.

[8] A. Atamtürk and D. Rajan. On splittable and unsplittable capacitated network design arc-set polyhedra. *Mathematical Programming*, 92:315–333, 2002.

[9] E. Balas. Facets of the knapsack polytope. *Mathematical Programming*, 8:146–164, 1975.

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

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

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

[13] M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Maritime transportation. In C. Barnhart and G. Laporte, editors, *Transportation: Handbooks of Transportation and Management Science*, volume 14, pages 189–284. North-Holland, 2007.

[14] J.-F. Cordeau, F. Pasin, and M. Solomon. An integrated model for logistics network design. *Annals of Operations Research*, 144:59–82, 2006.

[15] J.-F. Cordeau, P. Toth, and D. Vigo. A survey of optimization models for train routing and scheduling. *Transportation Science*, 32:380–404, 1998.

[16] G. Cornuéjols, R. Sridharan, and J.M. Thizy. A comparison of heuristics and relaxations for the capacitated plant location problem. *European Journal of Operational Research*, 50:280–297, 1991.

[17] A.M. Costa, J.F. Cordeau, and B. Gendron. Benders, metric and cutset inequalities for multicommodity capacitated network design. *Computational Optimization and Applications*, 42:371–392, 2009.

[18] T. G. Crainic, M. Gendreau, and J.M. Farvolden. A simplex-based tabu search method for capacitated network design. *INFORMS Journal on Computing*, 12:223–236, 2000.

[19] T.G. Crainic. Service network design in freight transportation. *European Journal of Operational Research*, 122:272–288, 2000.

[20] T.G. Crainic, A. Frangioni, and B. Gendron. Multicommodity capacitated network design. In P. Soriano and B. Sanso, editors, *Telecommunications Network Planning*, pages 1–19. Kluwer Academics Publisher, 1999.

[21] T.G. Crainic, A. Frangioni, and B. Gendron. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics*, 112:73–99, 2001.

[22] T.G. Crainic and M. Gendreau. Cooperative parallel tabu search for capacitated network design. *Journal of Heuristics*, 8:601–627, 2002.

[23] T.G. Crainic, B. Gendron, and G. Hernu. A slope scaling/Lagrangean perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design. *Journal of Heuristics*, 10:525–545, 2004.

[24] T.G. Crainic and K.H. Kim. Intermodal transportation. In C. Barnhart and G. Laporte, editors, *Transportation: Handbooks of Transportation and Management Science*, volume 14, pages 467–537. North-Holland, 2007.

[25] T.G. Crainic, N. Ricciardi, and G. Storchi. Models for evaluating and planning city logistics systems. *Transportation Science*, 43:432–454, 2009.

[26] A. Frangioni. `http://www.di.unipi.it/~frangio`, September 2012.

[27] V. Gabrel, A. Knippel, and M. Minoux. Exact solution of multicommodity network optimization problems with general step cost functions. *Operations Research Letters*, 25:15–23, 1999.

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

[29] B. Gendron and T.G. Crainic. Relaxations for multicommodity capacitated network design problems. Technical report, Publication CRT-945, Centre de recherche sur les transports, Université de Montréal, 1994.

[30] B. Gendron and F. Semet. Formulations and relaxations for a multi-echelon capacitated location-distribution problem. *Computers and Operations Research*, 36:1335–1355, 2009.

[31] I. Ghamlouche, T.G. Crainic, and M. Gendreau. Cycle-based neighbourhoods for fixed charge capacitated multicommodity network design. *Operations Research*, 51:655–667, 2003.

[32] I. Ghamlouche, T.G. Crainic, and M. Gendreau. Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. *Annals of Operations Research*, 131:109–133, 2004.

[33] Z. Gu, G.L. Nemhauser, and M.W.P. Savelsbergh. Lifted cover inequalities for 0-1 integer programs: computation. *INFORMS Journal on Computing*, 10:427–437, 1998.

[34] Z. Gu, G.L. Nemhauser, and M.W.P. Savelsbergh. Lifted cover inequalities for 0-1 integer programs: complexity. *INFORMS Journal on Computing*, 11:117–123, 1999.

[35] Z. Gu, G.L. Nemhauser, and M.W.P. Savelsbergh. Lifted flow cover inequalities for mixed 0-1 integer programs. *Mathematical Programming*, 85:439–467, 1999.

[36] P.L. Hammer, E.L. Johnson, and U.N. Peled. Facets of regular 0-1 polytopes. *Mathematical Programming*, 8:179–206, 1975.

[37] M. Hewitt, G.L. Nemhauser, and M.W.P. Savelsbergh. Combining exact and heuristic approches for the capacitated fixed-charge network flow problem. *INFORMS Journal on Computing*, 22:314–325, 2010.

[38] K. Holmberg and D. Yuan. A Lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Operations Research*, 48:461–481, 2000.

[39] N. Katayama, M. Chen, and M. Kubo. A capacity scaling heuristic for the multicommodity capacitated network design problem. *Journal of Computational and Applied Mathematics*, 232:90–101, 2009.

[40] G. Kliewer and L. Timajev. Relax-and-cut for capacitated network design. In *Proceedings of Algorithms-ESA 2005: 13th Annual European Symposium on Algorithms*, pages 47–58. Lecture Notes in Computer Science 3369, 2005.

[41] J.M.Y. Leung and T.L. Magnanti. Valid inequalities and facets of the capacitated plant location problems. *Mathematical Programming*, 44:271–291, 1989.

[42] Q. Louveaux and L.A. Wolsey. Lifting, superadditivity, mixed integer rounding and single node flow sets revisited. *Annals of Operations Research*, 153:47–77, 2007.

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

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

[45] T.L. Magnanti and R.T. Wong. Network design and transportation planning: models and algorithms. *Transportation Science*, 18:1–55, 1984.

[46] H. Marchand, A. Martin, R. Weismantel, and L.A. Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123:397–446, 2002.

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

[48] S. Martello and P. Toth. Upper bounds and algorithms for hard 0-1 knapsack problems. *Operations Research*, 45:768–778, 1997.

[49] M. Minoux. Network synthesis and optimum network design problems: models, solution methods and applications. *Networks*, 19:313–360, 1989.

[50] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, 1998.

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

[52] F. Ortega and L.A. Wolsey. A branch-and-cut algorithm for the single commodity uncapacitated fixed charge network flow problem. *Networks*, 41:143–158, 2003.

[53] M.W. Padberg, T.J. Van Roy, and L.A. Wolsey. Valid linear inequalities for fixed charge problems. *Operations Research*, 33:842–861, 1985.

[54] C. Raack, A.M.C.A. Koster, S. Orlowski, and R. Wessäly. On cut-based inequalities for capacitated network design polyhedra. *Networks*, 57:141–156, 2011.

[55] R.L. Rardin and L.A. Wolsey. Valid inequalities and projecting the multicommodity extended formulation for uncapacitated fixed charge network flow problems. *European Journal of Operational Research*, 71:95–109, 1993.

[56] I. Rodríguez-Martín and J. J. Salazar-González. A local branching heuristic for the capacitated fixed-charge network design problem. *Computers & Operations Research*, 37:575–581, 2010.

[57] M. Sellmann, G. Kliewer, and A. Koberstein. Lagrangian cardinality cuts and variable fixing for capacitated network design. In *Proceedings of Algorithms-ESA 2002: 10th Annual European Symposium on Algorithms*, pages 845–858. Lecture Notes in Computer Science 2461, 2002.

[58] J.I.A. Stallaert. The complementary class of generalized flow cover inequalities. *Discrete Applied Mathematics*, 77:73–80, 97.

[59] T.J. Van Roy and L.A. Wolsey. Solving mixed integer programming problems using automatic reformulation. *Operations Research*, 35:45–57, 1987.

[60] C. Vidal and M. Goetschalckx. A global supply chain model with transfer pricing and transportation cost allocation. *European Journal of Operational Research*, 129:134–158, 2001.

[61] L.A. Wolsey. Faces of linear inequalities in 0-1 variables. *Mathematical Programming*, 8:165–178, 1975.

[62] J. Zhu, T.G Crainic, and M. Gendreau. Scheduled Service Network Design for Freight Rail Transportation. *Operations Research*, 62(2):383–400, 2014.