CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

A Cutting-Plane Algorithm for Multicommodity Capacitated Fixed-Charge Network Design

Mervat Chouman
Teodor Gabriel Crainic
Bernard Gendron

May 2009

CIRRELT-2009-20

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince, local 2642
Québec (Québec)
Canada G1K 7P4
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Université Laval · UQÀM · HEC Montréal · École Polytechnique Montréal · Université de Montréal

A Cutting-Plane Algorithm for Multicommodity Capacitated Fixed-Charge Network Design

Mervat Chouman[1,2], Teodor Gabriel Crainic[1,2,*], Bernard Gendron[1,3]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Department of Management and Technology, Université du Québec à Montréal, C.P. 8888, succursale Centre-ville, Montréal, Canada H3C 3P8

[3] Department of Computer Science and Operations Research, Université de Montréal, C.P. 6128, succursale Centre-Ville, Montréal, Canada H3C 3J7

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 valid inequalities: the strong, cover, minimum cardinality, flow cover, and flow pack inequalities, the last four being expressed in terms of cutsets of the network. 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.

Keywords. Multicommodity capacitated fixed-charge network design, cutting-plane algorithm, valid inequalities, separation, lifting.

Acknowledgements. Partial funding for this project has been provided by the Natural Sciences and Engineering Research 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 de recherche sur la nature et les technologies (FQRNT) 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 experimentations.

Results and views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect those of CIRRELT.

Les résultats et opinions contenus dans cette publication ne reflètent pas nécessairement la position du CIRRELT et n'engagent pas sa responsabilité.

---

* Corresponding author: Teodor-Gabriel.Crainic@cirrelt.ca

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2009

© Copyright Chouman, Crainic, Gendron and CIRRELT, 2009

1 Introduction

Network design models are used in many applications, most notably in the fields of logistics, transportation and telecommunications [12, 31, 33]. In this paper, we study a particular case, the multicommodity capacitated fixed-charge network design problem (MCND), which can be described as follows. 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 (or origin-destination pairs) K to be routed according to a known demand 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 per unit of commodity k on arc (i, j) is denoted $c_{ij}^k \geq 0$, while the fixed design cost for arc (i, j) is denoted $f_{ij} \geq 0$. An origin $O(k)$ and a destination $D(k)$ are associated to each commodity k. The MCND is NP-hard since it contains as a special case the uncapacitated fixed charge network design problem, which is NP-hard as well [31].

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:

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

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

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

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

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

where $N_i^+ = \{j \in N | (i, j) \in A\}$ and $N_i^- = \{j \in N | (j, i) \in A\}$.

Constraints (2) correspond to flow conservation equations for each node and each commodity. Relations (3) represent capacity constraints for each arc. They also link together flow and design variables by forbidding any flow to circulate through 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 tools to solve such models. Here, however, the LP relaxation generally provides a weak lower bound [13]. Alternative relaxation approaches have been devised, in particular Benders decomposition [10] and Lagrangian-based procedures [13, 14, 19, 20, 26, 28, 37]. Heuristic methods have also been proposed for computing

feasible solutions [11, 15, 16, 21, 22]. In this paper, we present a cutting-plane method for computing better lower bounds, motivated by numerous successful applications of this methodology to other network design problems, closely related to the MCND [1, 2, 4, 5, 7, 8, 9, 17, 18, 29, 30, 35].

The cutting-plane method we propose is based on five classes of 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 of the network. These classes of inequalities have been applied to similar network design problems in the past, but not the MCND, to the best of our knowledge. Also, the efficient generation of these inequalities within a cutting-plane framework is a challenging task. Our contribution is threefold:

- We develop efficient separation and lifting procedures adapted to the MCND. In particular, we present a new separation procedure for flow cover and flow pack inequalities.
- Embedded within the cutting-plane framework, we develop several procedures for generating cutsets of the network, in particular a method inspired by metaheuristic approaches. These cutset generation procedures can be adapted to other network design problems [5, 35].
- We perform an extensive set of computational experiments that show the efficiency of our separation, lifting and cutset generation methods. In particular, we show that our implementation of cover and flow cover inequalities outperforms that of the state-of-the-art software CPLEX (version 10.1) [27] on our class of problem instances. When embedded in the B&B algorithm of CPLEX, we also show that our cutting-plane procedure allows to prove optimality for a majority of the instances, while most unsolved instances show optimality gaps within 3% when stopped after a reasonable CPU time limit.

The paper is organized as follows. The five classes of valid inequalities are described in Section 2. The separation and lifting procedures for these inequalities are presented in Section 3. The cutting-plane algorithm, including the cutset generation procedures, 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 Valid Inequalities

In this section, we present the five classes of valid inequalities that are used in our cutting-plane algorithm. The last four classes of valid inequalities are defined in terms of cutsets of the network.

2.1 Strong Inequalities

The strong inequalities (SI) are defined as follows:

$$x_{ij}^k \leq d^k y_{ij}, \quad \forall (i,j) \in A, k \in K. \tag{6}$$

Adding the SI to the model significantly improves the quality of the LP lower bounds [13, 19]. Although there is a polynomial number of SI ($|A||K|$), adding all of them to the LP relaxation yields very large models that frequently exhibit degeneracy. Only a small fraction of SI will be added within our cutting-plane algorithm.

2.2 Cover 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 cutset by $(S, \bar{S})$, i.e., the set of arcs that connect a node in S to a node in $\bar{S}$. The following *cutset inequality* is valid:

$$\sum_{(i,j) \in (S,\bar{S})} u_{ij} y_{ij} \geq d_{(S,\bar{S})}, \tag{7}$$

where $d_{(S,\bar{S})}$ is a lower bound on the amount of flow that must circulate across the cutset in any feasible solution. A simple way to compute $d_{(S,\bar{S})}$ is to use $K(S,\bar{S}) \subseteq K$, the set of commodities having their origin in S and their destination in $\bar{S}$, and to set $d_{(S,\bar{S})} = \sum_{k \in K(S,\bar{S})} d^k$. Since it is obtained from a linear combination of constraints (2) and (3), the cutset inequality is redundant for the LP relaxation of the MCND. It can still be useful, however, as some VI derived from it might improve the LP relaxation.

By complementing the y variables (replacing y_{ij} by $1 - y_{ij}$) in the cutset inequality, we obtain a classical 0-1 knapsack structure. The well-known cover inequalities (CI) for that structure [6, 25, 40] are based on the following definition (for the sake of clarity, we adapt to the MCND the classical terminology related to the 0-1 knapsack structure).

Definition 1: A set $C \subseteq (S,\bar{S})$ is a cover if the total capacity of the arcs in $(S,\bar{S}) \backslash C$ does not cover the demand:

$$\sum_{(i,j) \in (S,\bar{S}) \backslash C} u_{ij} < d_{(S,\bar{S})}.$$

Moreover, the cover $C \subseteq (S, \bar{S})$ is minimal if it is sufficient to open any arc in C to cover the demand:

$$\sum_{(i,j)\in(S,\bar{S})\setminus C} u_{ij} + u_{pq} \;\geq\; d_{(S,\bar{S})}, \forall (p,q) \in C.$$

For every cover $C \subseteq (S, \bar{S})$, the cover inequality (CI)

$$\sum_{(i,j)\in C} y_{ij} \;\geq\; 1 \tag{8}$$

is valid for the MCND. The basic idea behind this inequality is that one has to open at least one arc from the set C in order to meet the demand. In addition, it has been proven [6, 40] that if C is a minimal cover, we can apply a lifting procedure to derive a facet of the convex hull of the 0-1 knapsack structure defined by the cutset inequality for a given $(S, \bar{S})$.

2.3 Minimum Cardinality Inequalities

Let us assume the capacities of the arcs in $(S, \bar{S})$ are represented in non-increasing order: $u_{ij_{(t)}} \geq u_{ij_{(t+1)}}, t = 1, ..., |(S, \bar{S})| - 1$. We can then define the least number of arcs in $(S, \bar{S})$ that must be used in every feasible solution: $l_S = \max\left\{h | \sum_{t=1,...,h} u_{ij_{(t)}} < d_{(S,\bar{S})}\right\} + 1$. From this number, we can derive the minimum cardinality inequality (MCI), defined as:

$$\sum_{(i,j)\in(S,\bar{S})} y_{ij} \;\geq\; l_S. \tag{9}$$

This inequality has been used to strengthen relaxation bounds for the 0-1 knapsack problem [32].

2.4 Flow Cover Inequalities

To define the next classes of inequalities, we introduce the following notation. For any $L \subseteq K$, let $x^L_{ij} = \sum_{k\in L} x^k_{ij}$, $b^L_{ij} = \min\{u_{ij}, \sum_{k\in L} d^k\}$ and $d^L_{(S,\bar{S})} = \sum_{k\in K(S,\bar{S})\cap L} d^k$, for a given cutset $(S, \bar{S})$. 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 follows:

$$\begin{aligned}\sum_{(i,j)\in C_1} (x^L_{ij} + (b^L_{ij} - \mu)^+(1 - y_{ij})) \;\leq\; & \sum_{(j,i)\in D_2} \min\{b^L_{ji}, \mu\} y_{ji} + \sum_{(j,i)\in C_2} b^L_{ji} \\ & + d^L_{(S,\bar{S})} + \sum_{(j,i)\in(\bar{S},S)\setminus C_2\cup D_2} x^L_{ji},\end{aligned} \tag{10}$$

where $a^+ = \max\{0, a\}$ and $D_2 \subset (\bar{S}, S)\backslash C_2$. This inequality has been studied by several authors [24, 36, 39] and is implemented in state-of-the-art MIP software tools, such as CPLEX [27].

2.5 Flow Pack Inequalities

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 as follows [3, 38]:

$$\sum_{(i,j)\in C_1} x^L_{ij} + \sum_{(i,j)\in D_1} (x^L_{ij} - \min\{b^L_{ij}, -\mu\} y_{ij})) \leq - \sum_{(j,i)\in C_2} (b^L_{ji} + \mu)^+ (1 - y_{ji}) + \sum_{(j,i)\in(\bar{S},S)\backslash C_2} x^L_{ji} + \sum_{(i,j)\in C_1} b^L_{ij}. \quad (11)$$

where $D_1 \subset (S, \bar{S})\backslash C_1$.

3 Separation and Lifting Methods

In this section, we present separation and lifting procedures for each class of valid inequalities 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. We now turn to the cover and minimum cardinality inequalities, for which we present a general lifting procedure. In a subsequent section, we explain how we generate flow cover and flow pack inequalities using a new separation routine for these classes of inequalities.

3.1 Cover and Minimum Cardinality Inequalities

For both the cover and minimum cardinality inequalities, we assume a cutset $(S, \bar{S})$ is given (see Section 4 for a description of cutset generation procedures). To generate both types of 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 propose the following procedure, called **OpenCloseArcs**, which uses the variables T, the residual capacity ($\sum_{(i,j)\in(S,\bar{S})\backslash(C_1\cup C_0)} u_{ij}$), and D, the residual demand ($d_{(S,\bar{S})} - \sum_{(i,j)\in C_1} u_{ij}$). 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_0) and such that the residual capacity $(T - u_{ij})$ still covers the residual demand D. Similarly, the procedure attempts to open an arc (i, j) with a large value $\bar{y}_{ij}$ (as measured by a threshold ϵ_1) and such that there is still some residual demand to cover $(D - u_{ij} > 0)$. As in Gu *et al.* [23], the sets C_1 and C_0 can be derived from the variables having integer values at the current LP solution, by using ϵ_0 arbitrarily close to 0 and ϵ_1 arbitrarily close to 1. The outline of the procedure is summarized in Algorithm 1.

Algorithm 1 OpenCloseArcs

```
Initialize: T ← Σ_{(i,j)∈(S,S̄)} u_ij, D ← d_(S,S̄)
for arc (i, j) ∈ (S, S̄) (in arbitrary order) do
   if (ȳ_ij < ε_0) and (T − u_ij ≥ D) then
      Add (i, j) to C_0
      Close (i, j) by setting T ← T − u_ij
   end if
   if (ȳ_ij > ε_1) and (D − u_ij > 0) then
      Add (i, j) to C_1
      Open (i, j) by setting D ← D − u_ij and T ← T − u_ij
   end if
end for
```

Once the sets C_1 and C_0 are obtained, we define the restricted cutset inequality induced by C_1 and C_0 as

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

To define a cover C for this restricted cutset inequality, we have implemented the heuristic approach proposed by Gu *et al.* [23, 24] 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 (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 with this heuristic, it is easy to extract a minimal cover from it, by removing some of the arcs from the cover until the condition in Definition 1 is satisfied. Once the cover C is constructed, the induced inequality might be strengthened by the lifting procedure to be 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 used in the set $(S, \bar{S})\setminus(C_1 \cup C_0)$. 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 valid inequality.

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 the MCND.

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 the MCND. 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 value. 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. It is however possible to generate two different cover and minimum cardinality inequalities, one being violated by the current LP solution, while the other is not. Consequently, we incorporated both classes of inequalities into our cutting-plane procedure.

3.2 Flow Cover and Flow Pack Inequalities

To generate flow cover and flow pack inequalities, we use two simpler valid inequalities. First, the *single-arc flow pack inequality* (SFPI) is defined as follows:

$$\sum_{(i,j)\in C_1} x^L_{ij} + x^L_{rt} \leq (\sum_{(j,i)\in C_2} b^L_{ji} + d^L_{(S,\bar{S})})y_{rt} + \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x^L_{ji} + (1-y_{rt}) \sum_{(i,j)\in C_1} b^L_{ij}, \quad (12)$$

where $(r,t) \in (S,\bar{S})$, $C_1 \subseteq (S,\bar{S})\setminus\{(r,t)\}$ and $C_2 \subseteq (\bar{S},S)$. It is easy to show directly that this inequality is valid for the MCND. More interestingly, a necessary condition for the SFPI to be violated is $\mu < 0$, which is precisely what defines a flow pack (hence, its name). Moreover, the SFPI can be seen as a special case of the FPI, by letting $D_1 = \{(r,t)\}$ in the definition of the FPI. Indeed, if we assume $b^L_{rt} \geq -\mu$ and $b^L_{ji} \leq -\mu$, for all $(j,i) \in C_2$, then the FPI (11) reduces to:

$$\begin{aligned}
\sum_{(i,j)\in C_1} x^L_{ij} + x^L_{rt} &\leq -\mu y_{rt} + \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x^L_{ji} + \sum_{(i,j)\in C_1} b^L_{ij} \\
&= (\sum_{(j,i)\in C_2} b^L_{ji} + d^L_{(S,\bar{S})} - \sum_{(i,j)\in C_1} b^L_{ij})y_{rt} + \\
&\quad \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x^L_{ji} + \sum_{(i,j)\in C_1} b^L_{ij} \\
&= (\sum_{(j,i)\in C_2} b^L_{ji} + d^L_{(S,\bar{S})})y_{rt} + \\
&\quad \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x^L_{ji} + (1-y_{rt}) \sum_{(i,j)\in C_1} b^L_{ij},
\end{aligned}$$

which is precisely the SFPI (12).

The second valid inequality is called the *single-arc flow cover inequality* (SFCI):

$$\sum_{(i,j)\in C_1} x^L_{ij} + x^L_{rt} \leq (\sum_{(j,i)\in C_2} b^L_{ji} + d^L_{(S,\bar{S})})(1-y_{rt}) + \sum_{(j,i)\in(\bar{S},S)\setminus C_2} x^L_{ji} + y_{rt} \sum_{(i,j)\in C_1} b^L_{ij}, \quad (13)$$

where $(r,t) \in (\bar{S},S)$, $C_1 \subseteq (S,\bar{S})$ and $C_2 \subseteq (\bar{S},S)\setminus\{(r,t)\}$. Similarly to the SFPI, we can show the validity of the SFCI either directly, or by reducing it to a special case of the FCI when $\mu > 0$.

The interest of these single-arc inequalities is that their separation problems are simple once we fix the set of commodities L, in constrast with the FCI and the FPI, which remain hard to separate, even if we fix L. Indeed, given $(\bar{x},\bar{y})$ the current LP solution, sets $S \subset N$ and $L \subseteq K$, and an arc $(r,t) \in (S,\bar{S})$, separating the SFPI consists in setting

$$C_1 = \{(i,j) \in (S,\bar{S})\setminus\{(r,t)\} | \bar{x}^L_{ij} > (1-\bar{y}_{rt})b^L_{ij}\},$$

$$C_2 = \{(j,i) \in (\bar{S},S) | b^L_{ji}\bar{y}_{rt} < \bar{x}^L_{ji}\}.$$

For each cutset generated by the cutting-plane algorithm, the separation procedure thus scans each arc, trying to find violated SFPI associated to this arc. If the cutset is a singleton that is 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 (12) 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, then there are two cases: 1) $\mu < 0$; 2) $\mu + b^L_{rt} > 0$. In case 1), we lift the inequality to obtain a FPI. First, we initialize $D_1 = \{(r,t)\}$ and then add to D_1 each arc $(i,j) \in (S,\bar{S})\backslash C_1$ such that $\bar{x}^L_{ij} - \min\{b^L_{ij}, -\mu\}\bar{y}_{ij} > 0$. Then, we further lift the resulting FPI inequality by applying the lifting function proposed by Atamtürk [3]: we lift all variables in C_1 and the variables in $(\bar{S},S)\backslash C_2$ such that $\bar{y}_{ij} \leq \epsilon$. In case 2), we first add (r,t) to C_1 and recompute $\mu = \mu + b^L_{rt}$. 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 $b^L_{ij}(1 - y_{ij})$. We then set $D_2 = \{(j,i) \in (\bar{S},S)\backslash C_2 : \bar{x}^L_{ji} > \min\{b^L_{ji}, \mu\}\bar{y}_{ji}\}$. Finally, we lift the resulting FCI by applying the lifting function proposed by Atamtürk [3]: we lift all variables in C_2 and the variables in $(S,\bar{S})\backslash C_1$ such that $\bar{y}_{ij} \leq \epsilon$.

We proceed similarly when a violated SFCI is generated. There are again two cases: a) $\mu > 0$; b) $\mu - b^L_{rt} < 0$. In case a), we proceed as in case 2) above to obtain a lifted FCI. In case b), we first add (r,t) to C_2 and recompute $\mu = \mu - b^L_{rt}$. Then, we proceed as in case 1) 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, we apply well-known lifting strategies to generate a FCI or 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 [34]. 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 starts by solving the LP relaxation of formulation (1)-(5), 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 valid inequalities 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 fam-

ily 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 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 a parameter T_{max} to bound T, the number of calls to Phase II. The cutting-plane algorithm thus follows the general structure outlined in Algorithm 2, where δ (we use 0.1 in all our tests) and $\mathbf{M}_{max}$ are user-supplied parameters.

We now describe the three approaches used to generate families of cutsets in Phase II (Step 20 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. This approach is not meant to be efficient, but it is used as a basis of comparison for the two other approaches.

The second approach uses the notion of articulation set, which is a set $S \subset N$ such that removal of S disconnects, for at least one commodity k, its origin $O(k)$ from its destination $D(k)$. These are the only sets which might lead to non-trivial cutset inequalities based on $(S, \bar{S})$. In this so-called *Articulation* approach, we thus generate all cutsets $(S, \bar{S})$ where S is an articulation set.

In the first two approaches, the families of cutsets are generated only once, before the first execution of the **for** loop at step 19, and the corresponding cutsets are stored in memory for subsequent calls to Phase II. The third approach is significantly different, since 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 valid inequalities.

This approach is inspired by principles derived from metaheuristics. First, it calls a *construction* procedure to provide an initial partition of N into subsets of cardinality $\mathbf{M}$. 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 $\mathbf{M}$. 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

Algorithm 2 CuttingPlane

1: `Initialization`: Solve the weak relaxation; let z and $\bar{y}$ be the optimal value and design solution, respectively
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; let z and $\bar{y}$ be the optimal value and design solution, respectively
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** $T \geq T_{max}$ **then**
16: stop
17: **end if**
18: $z_{last} \leftarrow z$ and $T \leftarrow T + 1$
19: **for** $\mathbf{M} = 2$ to $\mathbf{M}_{max}$ **do**
20: Generate a family of cutsets based on subsets of N of cardinality $\mathbf{M}$
21: Generate cuts, using the current family of cutsets
22: **if** some cuts were found **then**
23: Solve the LP relaxation; let z and $\bar{y}$ be the optimal value and design solution, respectively
24: **if** $\bar{y}$ is integral **then**
25: stop
26: **end if**
27: **end if**
28: **end for**
29: **if** $z - z_{last} > \delta$ **then**
30: go to 6
31: **end if**

corresponding family of cutsets. To summarize, in the *Heuristic* approach, the family of cutsets generated at Step 20 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 $\mathbf{M}$ is obtained by the construction procedure called **GenerateMultiSet**(M). 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 $\mathbf{M}$. 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. The outline of the procedure is summarized in Algorithm 3.

Algorithm 3 GenerateMultiSet(M)

1: `Initialize`: $\bar{N} \leftarrow N$, $l \leftarrow 1$
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 valid inequalities that are aggregations of previously generated valid inequalities. Figure 1 illustrates such a situation, where we assume that all possible violated cuts using cutsets induced by subsets of cardinality 1 have already been generated. Currently, the procedure is constructing subsets of cardinality 2. If we chose to include the free node 2 into the set $S_l = \{1\}$ to create the new set $S_l = \{1, 2\}$, we would identify a new cutset, but the resulting cutset inequality

would just be the aggregation of the previously generated cutset inequalities induced by $\{1\}$ and $\{2\}$. Thus, this new cutset would not identify any new cuts.

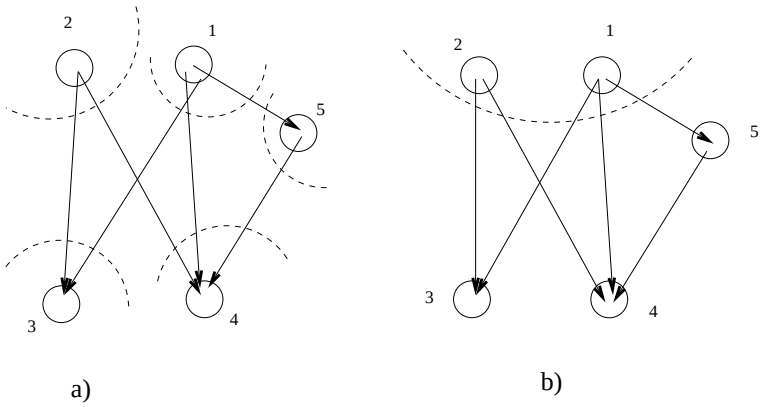

Figure 1: GenerateMultiSet Example

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**$((S_l)_{l=1,\ldots,L}, W, W_N)$. 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 $n \in N \setminus W_N$ 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 valid inequalities 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. This is illustrated in Figure 2, where the procedure stops after successively moving nodes from S_2 to S_1, from S_3 to S_2, from S_4 to S_3, and from S_1 to S_4. By doing so, all subsets have the same cardinality as before the exchanges. The outline of this procedure is presented in Algorithm 4.

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

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

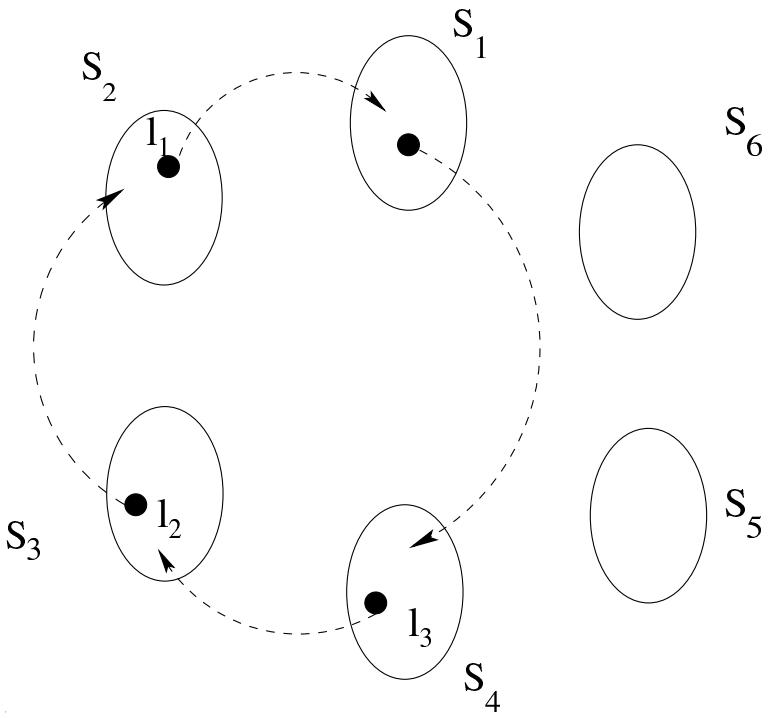

Figure 2: MultiExchange Example

5 Computational Results

Computational experiments were performed with five objectives in mind:

- Calibrate the parameters of the procedures.
- Compare the performance of our separation and lifting procedures for cover and flow cover inequalities with those of the state-of-the-art software CPLEX (version 10.1).
- Compare the relative performance of the different classes of valid inequalities.
- Test the performance of the cutset generation procedures.
- Evaluate the quality of the formulations obtained from different variants of the cutting-plane algorithm, by performing a state-of-the-art B&B algorithm on each of these formulations.

Following a preliminary section that describes the data instances and the performance measures used in the experiments, we present and analyze the results in the five subsequent subsections, each dedicated to one of the objectives stated above.

5.1 Data Instances and Performance Measures

Computational experiments were conducted on a set of 196 instances used in several papers on the MCND problem and described in detail by Crainic *et al.* [14]. These problem instances consist of general transshipment networks with one commodity per origin-destination and no parallel arcs. Associated to each arc are three positive quantities: the capacity, the fixed charge, and the transportation cost (the same for all commodities). 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 consists of 31 problem instances with many commodities compared to the number of nodes, while Class II contains 12 problem instances with few commodities compared to the number of nodes. In addition to the numbers of nodes, arcs, and commodities, each instance in Classes I and II is characterized by two more letters indicating the fixed cost level compared to the transportation cost, "F" for high and "V" for low, and the capacity level compared to the total demand, "T" for tight and "L" for loose. Class III is divided into two categories, A and B, each containing nine sets of nine problem instances each (with three exceptions corresponding to the smaller-network instances). 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. Thus, $F = 0.01$ (F01), $F = 0.05$ (F05), and $F = 0.1$ (F10) are used to qualify the fixed costs relative to the transportation costs, where the fixed-cost ratio is computed as $F = |K| \sum_{(i,j)\in A} f_{ij} / \sum_{k\in K} d^k \sum_{(i,j)\in A} c^k_{ij}$. Similarly, $C = 1$ (C1), $C = 2$ (C2), and $C = 8$ (C8) are used to qualify the tightness of the total capacity compared to the total demand, where the capacity ratio is computed as $C = |A| \sum_{k\in K} d^k / \sum_{(i,j)\in A} u_{ij}$. Class III-A contains 72 small size problem instances with 10 nodes, while Class III-B contains 81 medium to large size instances with 20 nodes.

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

- The time, t, to compute the lower bound, where all experiments are performed on a Sun Entreprise 10000 with 64 Gigabytes of RAM operating under Solaris 2.7. The procedures are coded in C++. To solve the LP relaxations, we use the dual simplex implementation in CPLEX [27]. We use the time, t^w, to compute the weak relaxation bound as a basis of comparison. To compare the time t required by the cutting-plane algorithm with t^w, we use the measure
$$\Delta t^w = \frac{100(t - t^w)}{t^w}.$$
- The gap between the lower bound and the value of a reference solution. For the weak relaxation, we use as reference solution the best known feasible solution of

Description	Nb	Weak LP Δz^*	Weak LP t^w	Description	Nb	Weak LP Δz^*	Weak LP t^w
Class I				Class II			
20,230,40	(3)	7.70%	0.2	25,100,10	(3)	29.02%	0.1
20,230,200	(4)	28.33%	5.7	25,100,30	(3)	24.44%	0.3
20,300,40	(4)	9.74%	0.3	100,400,10	(3)	37.30%	1.1
20,300,200	(4)	21.41%	8.3	100,400,30	(3)	34.56%	1.5
30,520,100	(4)	18.45%	7.9				
30,520,400	(4)	15.62%	35.4				
30,700,100	(4)	17.72%	2.8				
30,700,400	(4)	18.07%	39.8				
Average	(31)	17.38%	12.3	Average	(12)	31.33%	0.8
Class III-A				Class III-B			
10,35,10	(6)	12.61%	0.0	20,120,40	(9)	21.93%	0.5
10,35,25	(6)	17.96%	0.1	20,120,100	(9)	19.57%	3.0
10,35,50	(6)	14.34%	0.1	20,120,200	(9)	16.71%	8.5
10,60,10	(9)	20.26%	0.0	20,220,40	(9)	29.93%	0.7
10,60,25	(9)	16.06%	0.1	20,220,100	(9)	26.95%	2.5
10,60,50	(9)	18.67%	0.2	20,220,200	(9)	24.05%	13.0
10,85,10	(9)	17.25%	0.0	20,320,40	(9)	32.34%	0.9
10,85,25	(9)	18.69%	0.1	20,320,100	(9)	30.56%	2.9
10,85,50	(9)	21.54%	0.2	20,320,200	(9)	28.10%	15.2
Average	(72)	17.80%	0.1	Average	(81)	25.57%	5.3

Table 1: Classes and Problem Dimensions

value z^*, which corresponds to the best (often optimal) solution obtained by using CPLEX [27] on the so-called *strong* formulation, obtained by adding all strong inequalities to the weak formulation. A limit of 10 hours of computation was imposed for each instance. The results show that CPLEX is quite efficient at solving small-size instances within the time allowed. However, when the problem size is increasing (especially the number of commodities), CPLEX is unable to prove optimality within the available computational time. For these instances, the best feasible solution found is then used. For the weak relaxation lower bound z^w, we thus report the following gap measure:

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

For the lower bound z computed by the cutting-plane procedure, the reference solution 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 among the classes with respect to problem dimension. 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 time required to solve the weak relaxation is given in column t^w. The "*Average*" line shows the gap average over all instances in each class along with the average time required to compute the bounds. The results in column Δz^* confirm the poor quality of the lower bounds generated by the weak relaxation.

5.2 Parameter Calibration

Preliminary tests to calibrate the parameters were performed over a subset of 38 instances with various characteristics selected from Classes I, II, and III-B. The chosen test set is shown in Table 2. These tests aim to calibrate the lifting tolerance parameters ϵ_0 and ϵ_1 used to generate CI and MCI in Algorithm 1, as well as ϵ used to generate FPI and FCI lifted inequalities. To facilitate the calibration and the comparison, we tested several values for ϵ while setting $\epsilon_0 = \epsilon$ and $\epsilon_1 = 1 - \epsilon$. Only Phase I of the cutting-plane algorithm was performed, i.e., only cutsets based on singletons were used. This choice is justified by the fact that most of the bound improvement is obtained in Phase I, the additional improvement provided by Phase II being relatively modest, as we will see in Section 5.5.

Classes	Dimension	Nb	Dimension	Nb	Dimension	Nb	Total
Class I	30,520,100	(4)	30,520,200	(4)			(8)
Class II	10,100,400	(3)					(3)
Class III-B	20,300,40	(9)	20,300,100	(9)	20,300,200	(9)	(27)
Total							(38)

Table 2: Problem Instances for Parameter Calibration

	CI		MCI		FCI		FPI	
ϵ	Δz^w	Δt^w	Δz^w	Δt^w	Δz^w	Δt^w	Δz^w	Δt^w
0	**8.10%**	5.2%	6.63%	4.9%	**36.72%**	96.3%	**37.26%**	111.7%
0.5	7.49%	5.0%	**7.70%**	5.2%	36.43%	98.7%	37.19%	126.8%
1	7.55%	5.2%	7.48%	5.2%	36.33%	112.0%	37.17%	181.5%

Table 3: Parameter Calibration

Table 3 displays the results obtained by using CI, MCI, FPI, and FCI, each family alone, in the cutting-plane algorithm. Three values for ϵ (0, 0.5, and 1) have been considered. Bold entries in Table 3 indicate the best results obtained for each family and each $\epsilon-$value. Therefore, $\epsilon = 0$ is the best value for CI, FPI, and FCI, while $\epsilon = 0.5$ is the best for MCI. These results are consistent with those obtained in the literature on CI [23], FCI and FPI [3] . For the MCI, the value of 0.5 is somewhat intuitive. Indeed, unlike the CI, which is based on a minimum cover, the MCI by itself is not strong since it is based on all arcs in the cutset. Therefore, closing and opening as many arcs as possible, as reflected by the value $\epsilon = 0.5$, and then performing lifting, will lead to a stronger inequality. Consequently, in the remaining tests, $\epsilon = 0$ is used for CI, FPI, and FCI, while $\epsilon = 0.5$ is used for MCI.

In addition to the best value for the lifting parameters, these preliminary results indicate that using CI or MCI provides somewhat a modest improvement of bound quality, about 8% for an extra 5% of computational time, when compared to the important improvement obtained from FPI and FCI, about 37% for an extra 100% of computational time. These results seem to point to the fact that inequalities based on both continuous and 0-1 variables are more effective than those based on 0-1 variables only.

5.3 Comparison with CPLEX Cuts

Table 4 displays the per-class average results obtained by the cutting-plane method implemented in CPLEX and those of our cutting-plane algorithm. We aim especially to compare the CPLEX implementation of CI and FCI with our own implementation for the same classes of valid inequalities. To make 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 valid inequalities in both methods, "*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 valid inequalities are used, but only single-node cutsets are used in the cutset generation procedure.

The results indicate that the cutting-plane algorithm we propose outperforms CPLEX in terms of bound quality and computational effort. Indeed, for a slightly longer time than CPLEX, our CI implementation obtains a gap improvement of 8.54% compared to 5.4% obtained by CPLEX. Moreover, for a substantially less computational effort (except for Class II instances), our FCI implementation obtains better bounds than CPLEX FCI. The most conclusive result is obtained when comparing the "*All*" and "*Enum1*" methods. For 1/6 of the computational effort needed by CPLEX, our "*Enum1*" method gets an extra 4.78% of bound gap improvement on average over the 196 instances. In fact, this extra gap improvement reaches up to 14.07% for Class II instances for a larger, but still reasonable computing time.

5.4 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 valid inequalities. As in the previous sections, 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 based on problem dimensions and characteristics.

CPLEX										
Classes		CI			FCI			All		
		Δz^w	Δt^w	Cuts	Δz^w	Δt^w	Cuts	Δz^w	Δt^w	Cuts
Class I	(31)	0.48%	1.4%	7	17.41%	194.1%	768	17.41%	202.4%	784
Class II	(12)	23.28%	0.6%	18	38.11%	5.3%	112	38.59%	5.0%	113
Class III-A	(72)	5.45%	0	6	17.82%	4.4%	77	17.83%	4.3%	75
Class III-B	(81)	4.60%	0.9%	14	27.91%	92.6%	358	27.90%	95.1%	357
Average	(196)	5.40%	0.6%	10	23.17%	70.9%	305	23.20%	72.5%	306
Cutting-Plane										
Classes		CI			FCI			Enum1		
		Δz^w	Δt^w	Cuts	Δz^w	Δt^w	Cuts	Δz^w	Δt^w	Cuts
Class I	(31)	1.00%	1.3%	21	17.29%	26.8%	1278	19.21%	18.1%	2858
Class II	(12)	24.38%	7.4%	28	50.71%	28.6%	706	52.66%	17.2%	1312
Class III-A	(72)	8.78%	0.5%	16	20.31%	3.4%	163	21.19%	3.3%	335
Class III-B	(81)	8.86%	1.2%	38	31.35%	51.8%	1114	33.70%	18.2%	2134
Average	(196)	8.54%	1.4%	27	26.25%	28.6%	766	27.98%	12.6%	1537

Table 4: CPLEX Cuts Versus Cutting-Plane Algorithm

Table 5 shows the improvement gap, Δz^w, and the additional computational time, Δt^w, 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 valid inequalities, except the one identifying the respective row. These results confirm 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. In terms of gap improvement, FPI slightly outperforms SI, but at the expense of a much larger computational effort. We also see that removing the generation of SI is detrimental to the performance of the cutting-plane algorithm, as the gap improvement then decreases by 1.01% with a significant increase in computational time. Removing any of the other classes of inequalities is also somewhat counter-productive, as it leads to reductions in the gap improvement without significant improvements in the computational time.

	None+		All-	
	Δz^w	Δt^w	Δz^w	Δt^w
$\emptyset$	0%	0%	27.98%	12.6%
SI	26.53%	7.3%	26.97%	30.0%
CI	8.54%	1.4%	27.92%	12.8%
MCI	8.00%	1.4%	27.97%	12.6%
FCI	26.25%	28.6%	27.97%	10.9%
FPI	26.75%	32.9%	27.94%	10.5%

Table 5: Comparison of Valid Inequalities

Table 6 shows a detailed analysis, according to problem dimension, of the effect of activating each class of inequalities alone. 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 much stronger than the other classes of inequalities. These results show an important gap improvement for a reasonable extra time for the different methods and different classes. Indeed, the per-class average gap improvement ranges from 17.29% up to 52.09%. Although all three classes provide a good gap improvement, SI shows the best

performance regarding the bound gap improvement and the extra time needed. However, SI is outperformed, in terms of bound gap, by FPI and FCI for Classes II and III-A. In fact, the best average improvement for these two classes is obtained by FPI, which reaches 52.09% for Class II and 20.68% for Class III-A and gets up to 69.13% for some difficult Class II instances (e.g., 100,400,30). It is important to note that instances in Classes II and III-A are characterized by a small number of commodities when compared to the number of nodes. On the contrary, when the number of commodities is significantly larger than the number of nodes, as for most instances in Classes I and III-B, SI outperforms FPI and FCI. Among these two classes of inequalities, FPI generally outperforms FCI in terms of bound improvement, for a similar computational effort.

Problems		SI			FCI			FPI		
		Δz^w	Δt^w	Cuts	Δz^w	Δt^w	Cuts	Δz^w	Δt^w	Cuts
Class I										
20,230,40	(3)	7.59%	2.4%	97	7.77%	4.3%	243	7.85%	3.3%	217
20,230,200	(4)	31.72%	14.7%	1310	27.35%	37.0%	1744	28.17%	36.6%	2288
20,300,40	(4)	9.88%	2.2%	107	10.01%	3.8%	254	10.09%	3.9%	270
20,300,200	(4)	23.24%	9.9%	910	20.65%	16.2%	1270	21.08%	15.5%	1646
30,520,100	(4)	21.58%	11.8%	603	20.54%	22.1%	1255	21.06%	20.0%	1464
30,520,400	(4)	16.75%	27.9%	1492	14.53%	42.6%	1946	14.92%	39.6%	2475
30,700,100	(4)	19.55%	12.2%	551	18.60%	21.6%	1112	19.16%	20.7%	1351
30,700,400	(4)	19.32%	35.7%	1705	16.52%	60.9%	2145	17.01%	54.9%	2697
Average	(31)	19.06%	15.0%	871	17.29%	26.8%	1278	17.73%	25.0%	1594
Class II										
25,100,10	(3)	33.16%	2.0%	68	38.54%	4.8%	212	39.51%	5.1%	258
25,100,30	(3)	33.85%	1.7%	122	34.86%	3.6%	261	35.73%	3.9%	308
100,400,10	(3)	48.56%	13.5%	168	62.85%	22.5%	858	64.00%	19.8%	898
100,400,30	(3)	63.76%	24.7%	477	66.61%	83.4%	1492	69.13%	79.2%	1562
Average	(12)	44.83%	10.5%	209	50.71%	28.6%	706	52.09%	27.0%	757
Class III-A										
10,35,10	(6)	11.36%	0.4%	17	11.76%	1.1%	42	11.76%	1.0%	41
10,35,25	(6)	20.94%	0.6%	53	21.41%	1.8%	94	21.53%	1.8%	108
10,35,50	(6)	16.90%	0.7%	113	16.94%	1.8%	119	17.18%	2.2%	150
10,60,10	(9)	22.69%	0.5%	26	24.98%	1.9%	87	25.50%	2.0%	97
10,60,25	(9)	16.00%	1.2%	74	17.04%	4.0%	180	17.51%	4.3%	220
10,60,50	(9)	21.55%	1.9%	148	20.53%	5.3%	239	21.19%	7.0%	337
10,85,10	(9)	17.11%	0.5%	23	19.28%	3.2%	132	19.72%	3.2%	131
10,85,25	(9)	19.71%	1.0%	68	21.37%	3.7%	208	21.65%	3.8%	225
10,85,50	(9)	26.32%	2.2%	146	25.87%	5.5%	291	26.20%	6.2%	339
Average	(72)	19.52%	1.1%	76	20.31%	3.4%	163	20.68%	3.7%	194
Class III-B										
20,120,40	(9)	23.12%	2.6%	167	24.21%	7.0%	422	24.88%	7.1%	488
20,120,100	(9)	23.53%	2.9%	435	22.03%	6.7%	560	22.56%	7.9%	731
20,120,200	(9)	20.36%	2.5%	817	18.30%	6.9%	550	18.42%	7.9%	729
20,220,40	(9)	40.87%	7.9%	339	40.58%	31.9%	835	41.02%	30.4%	966
20,220,100	(9)	34.05%	10.8%	607	31.80%	28.4%	1044	32.44%	28.2%	1297
20,220,200	(9)	29.68%	8.1%	938	26.64%	19.9%	1312	27.27%	30.0%	1661
20,320,40	(9)	47.66%	15.5%	553	47.41%	154.1%	1599	47.85%	211.6%	1878
20,320,100	(9)	41.91%	20.2%	967	39.79%	117.1%	1734	40.31%	153.3%	2198
20,320,200	(9)	35.06%	13.6%	1329	31.35%	93.8%	1973	31.81%	317.4%	2416
Average	(81)	32.92%	9.4%	684	31.35%	51.8%	1114	31.84%	62.6%	1374

Table 6: Comparison of Valid Inequalities with Respect to Problem Dimension

Table 7 summarizes the results of the same experiments, presented with respect to the

different levels of fixed-cost and capacity ratios. Two major conclusions emerge from these results. First, the gap improvement increases with the importance of fixed costs (for the same capacity level). Second, the gap improvement decreases as the capacities get tighter (for the same fixed-cost level). In terms of gap improvement, SI often outperforms the other classes for loose-capacity instances, while FPI usually outperforms the other classes for tight-capacity instances. In terms of computational time, SI clearly outperforms the other classes, irrespective of the problem characteristics. Not only the identification of violated valid inequalities is easier with SI than with FCI and FPI, but also the number of cuts generated by SI is significantly less than with the two other classes.

Problems			SI			FCI			FPI		
			Δz^w	Δt^w	Cuts	Δz^w	Δt^w	Cuts	Δz^w	Δt^w	Cuts
Class I											
V	L	(8)	19.79%	15.7%	843	17.85%	29.0%	1147	18.24%	25.2%	1415
V	T	(8)	13.76%	13.0%	655	12.55%	22.2%	1016	12.88%	20.0%	1199
F	L	(7)	25.76%	17.4%	1204	23.04%	35.0%	1748	23.69%	35.5%	2313
F	T	(8)	17.78%	14.0%	824	16.45%	21.9%	1262	16.85%	20.6%	1539
Average		(31)	19.06%	15.0%	871	17.29%	26.8%	1278	17.73%	25.0%	1594
Class II											
Problems			SI			FCI			FPI		
			Δz^w	Δt^w	Cuts	Δz^w	Δt^w	Cuts	Δz^w	Δt^w	Cuts
V	L	(4)	4.51%	14.7%	42	5.28%	15.7%	141	5.42%	15.3%	127
V	T	(4)	0.64%	11.8%	110	0.86%	12.5%	318	0.90%	12.5%	293
F	L	(4)	96.26%	13.3%	407	102.31%	58.8%	1120	105.00%	56.5%	1267
F	T	(4)	35.66%	4.8%	144	46.76%	12.9%	768	48.12%	10.6%	793
Average		(12)	44.83%	10.5%	209	50.71%	28.6%	706	52.09%	27.0%	757
Class III-A											
	C1	(9)	9.39%	0.6%	52	9.43%	1.3%	85	9.45%	1.4%	101
F01	C2	(9)	6.09%	0.7%	50	6.62%	1.7%	86	6.63%	1.7%	97
	C8	(6)	3.32%	0.5%	40	4.49%	2.4%	118	4.69%	2.9%	128
	C1	(9)	29.20%	1.2%	90	28.78%	3.2%	169	29.30%	3.2%	201
F05	C2	(9)	18.97%	1.1%	81	19.76%	3.0%	158	20.14%	2.9%	184
	C8	(6)	11.08%	0.6%	54	13.71%	4.1%	194	14.11%	4.4%	210
	C1	(9)	42.10%	2.0%	122	41.41%	5.0%	217	42.01%	7.1%	293
F10	C2	(9)	27.89%	1.5%	104	29.07%	5.0%	216	29.84%	5.5%	268
	C8	(6)	19.40%	0.9%	70	22.92%	4.9%	252	23.26%	4.9%	267
Average		(72)	19.52%	1.1%	76	20.31%	3.4%	163	20.68%	3.7%	194
Class III-B											
	C1	(9)	26.11%	6.8%	473	24.65%	15.9%	734	25.05%	16.8%	926
F01	C2	(9)	18.48%	5.6%	384	17.66%	11.6%	662	17.99%	10.7%	775
	C8	(9)	6.44%	2.0%	317	6.94%	5.8%	547	7.04%	5.7%	522
	C1	(9)	51.99%	21.2%	1109	48.41%	129.7%	1707	49.15%	100.3%	2163
F05	C2	(9)	36.02%	10.7%	697	34.27%	38.0%	1221	34.94%	42.1%	1550
	C8	(9)	18.53%	2.1%	434	18,60%	6.9%	785	18.85%	6.3%	785
	C1	(9)	64.89%	23.2%	1392	60.85%	196.2%	2073	61.63%	317.4%	2848
F10	C2	(9)	45.82%	10.5%	830	43.24%	54.5%	1413	44.14%	56.0%	1803
	C8	(9)	27.98%	2.0%	517	27.48%	7.2%	886	27,78%	8.3%	991
Average		(81)	32.92%	9.4%	684	31.35%	51,8%	1114	31.84%	62.6%	1374

Table 7: Comparison of Valid Inequalities with Respect to Problem Characteristics

5.5 Comparison of Cutset Generation Procedures

In this section, we compare the results obtained by 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 valid inequalities 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 have tested three values of j: 2, 3, and 4, but we report only the results obtained with the first two values. Indeed, *Enum4* runs out of memory for Class II instances, since enumerating all cutsets up to cardinality 4 for instances with 100 nodes leads to unrealistically large-size instances (4 087 975 cutsets). Moreover, the results of *Enum4* for the other classes show that the bound improvement is not significant, with an improvement of 0.01% on average for 10 times more computational effort when compared to *Enum3* results.
- *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. The parameters of the procedures were calibrated and the following values were used: $T_{max} = 5$, $\mathbf{M}_{max} = \left\lceil \frac{N}{3} \right\rceil$, 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.

Table 8 displays the average results obtained by these cutset generation methods. The results obtained by *Enum1* show that Phase I of the cutting-plane algorithm improves the weak relaxation bound by 27.98% on average for a relatively small increase in computational effort of 12.6%. By increasing the time by 14.9%, *Enum2* reaches an additional improvement of only 0.27% on average over the 196 problem instances. This additional improvement reaches 0.37% on average with *Enum3*, but the computational effort is then substantially more important. When comparing the results of *Artic* with those of *Enum2*, we see that the enumeration of articulation sets of cardinality 2 is almost as effective (in terms of the bounds) as the complete enumeration of all subsets of cardinality 2 and is slightly more efficient (in terms of computational time). Not surprisingly, the reduction of the computational effort increases with the number of nodes,

		Δz^w	Δt^w	Cuts	Δz^w	Δt^w	Cuts	Δz^w	Δt^w	Cuts
classes		Enum1			Enum2			Enum3		
Class I	(31)	19.21%	18.1%	2858	19.22%	30.7%	2868	19.23%	174.5%	2895
Class II	(12)	52.66%	17.2%	1312	54.62%	143.6%	1590	55.31%	3878.4%	2173
Class III-A	(72)	21.19%	3.3%	335	21.52%	8.2%	361	21.65%	23.6%	384
Class III-B	(81)	33.70%	18.1%	2134	33.78%	26.3%	2169	33.81%	93.3%	2195
Average	(196)	27.98%	12.6%	1537	28.25%	27.5%	1580	28.35%	312.3%	1639
classes		Artic			Heur			ArticHeur		
Class I	(31)	19.22%	30.6%	2873	19.23%	43.2%	2875	19.23%	55.7%	2880
Class II	(12)	54.25%	70.3%	1462	55.18%	113.4%	1653	55.24%	161.8%	1691
Class III-A	(72)	21.52%	7.9%	358	21.50%	25.7%	358	21.56%	29.5%	361
Class III-B	(81)	33.77%	26.4%	2164	33.80%	38.0%	2165	33.81%	45.9%	2175
Average	(196)	28.23%	23,0%	1570	28.28%	38.9%	1582	28.32%	48.5%	1591

Table 8: Comparison of Cutset Generation Procedures

the best results being obtained for Class II instances. Interesting results are obtained with the heuristic approaches *Heur* and *ArticHeur*, which show bound improvements and computational times that are competitive with those of the enumeration methods. Overall, the best bound improvements are obtained with *Enum3*, but with a prohibitive computational effort. A good tradeoff between bound quality and computational effort is obtained with *ArticHeur*. Even then, we note that the bound obtained after Phase II only slightly improves upon the bound computed after Phase I, with less than 0.5% on average, with the exception of Class II instance, for which the bound improvement reaches more than 2.5% on average.

5.6 Evaluation of Cutting-Plane Formulations

To evaluate more precisely the performance of the proposed cutting-plane algorithms, we perform the B&B algorithm of CPLEX [27] on the formulations obtained from *SI* (i.e., only the strong inequalities are generated by the cutting-plane algorithm), *Enum1*, and *ArticHeur*. A limit of 2 hours of CPU time is imposed for each instance. Moreover, the best known feasible solution for each instance is provided as the initial incumbent. This way, our experiments measure the quality of the lower bounds in terms of their ability to prune the search tree. The results are provided in Table 9, which is divided into two parts: "*Solved Problems*" and "*Unsolved Problems*." The former shows the results obtained for the problem instances that are solved to optimality within the time allowed using any of the three formulations. Of the 196 instances, there are 135 instances in this category. In "*Unsolved Problems*" we display the results obtained for problems that could not be solved using any of the three formulations. Of the remaining 61 instances, there are 58 instances in this category. Two of the remaining 3 instances have been solved using *SI* only, while the third has been solved using *Enum1* only.

First, we analyze the "*Solved Problems*." Column "*Solved*" shows the number of solved problems per class, while columns "*t*" and "*Nodes*" give the average time and

the average number of nodes required for each formulation. These results indicate the superiority of *ArticHeur* over *Enum1* and *SI* in terms of the number of nodes required to prove optimality. In particular, *ArticHeur* uses about 1/5 of the number of nodes required by *SI* for Class II instances. Despite this reduction, *SI* outperforms *ArticHeur* in terms of computational time, except for Class II instances for which the average time needed by both formulations is almost the same. Similarly, *Enum1* requires less nodes, but is also more time consuming per node when compared to *SI*.

Solved Problems								
Classes		Solved	SI		Enum1		ArticHeur	
			t	Nodes	t	Nodes	t	Nodes
Class I	(31)	10	698	3547	950	2312	984	2473
Class II	(12)	9	705	70129	522	25020	650	15156
Class III-A	(72)	72	2	215	3	171	3	160
Class III-B	(81)	44	247	2652	669	2168	554	1644
Average	(196)	135	180	5917	325	2637	299	1815
Unsolved Problems								
Classes		Unsolved	SI		Enum1		ArticHeur	
			Δz^*	Nodes	Δz^*	Nodes	Δz^*	Nodes
Class I	(31)	21	3.16%	7971	3.15%	5410	3.16%	4914
Class II	(12)	3	7.69%	38974	6.99%	10379	6.67%	5017
Class III-B	(81)	34	2.96%	13844	3.20%	6102	3.20%	5793
Average	(124)	58	3.28%	13017	3.38%	6073	3.36%	5434

Table 9: CPLEX B&B, 2 hours CPU Time Limit, Solved and Unsolved Problems

Next, we analyze the "*Unsolved Problems*." Column *Unsolved* shows the number of unsolved problems per class, while Δz^* shows the gap with respect to the best known solution z^*. These results indicate that all formulations are able to provide impressive optimality gaps on average. Although *SI* provides the best optimality gap for the 58 unsolved problems, the optimality gap for Class II instances is worse than the one obtained by *Enum1* and *ArticHeur* formulations. We further note that this gap improvement has been obtained within the same limited time and while exploring substantially less nodes. We also note that, by exploring only half of the nodes when compared to *Enum1*, *ArticHeur* provides better solutions for Class II instances. These results confirm those of the previous section, which have shown the superiority of the *ArticHeur* method when compared to the other methods for Class II instances.

We have performed additional experiments on the 61 instances that are not in the category of "*Solved Problems*." For this set of instances, Table 10 gives the results for solved and unsolved problems using the B&B algorithm of CPLEX applied to the three formulations with a limit of 10 hours of CPU time imposed for each instance. We note that, of the 61 instances, 12 instances fall in the "*Solved Problems*" category and 42 instances fall in the "*Unsolved Problems*" category. The remaining 7 instances have been solved by only one or two formulations. In particular, two of Class III instances have been solved solely by *SI*, while one Class I and one Class III instances have been solved

only by *Enum1* and *ArticHeur*. In general, for the 61 instances, the results indicate that, by allowing more time, the optimal gap is reduced and reaches close to 3% on average for all formulations.

Solved Problems								
Classes		Solved	SI		Enum1		ArticHeur	
			t	Nodes	t	Nodes	t	Nodes
Class I	(21)	2	11350	14222	10191	3462	11746	5135
Class II	(3)	0	-	-	-	-	-	-
Class III-B	(37)	10	12405	56077	15398	33348	15311	25798
Average	(61)	12	12229	49101	14531	28367	14717	22354
Unsolved Problems								
Classes		Unsolved	SI		Enum1		ArticHeur	
			Δz^*	Nodes	Δz^*	Nodes	Δz^*	Nodes
Class I	(21)	19	2.98%	25114	2.97%	15776	2.99%	14937
Class II	(3)	3	6.76%	178931	6.22%	52794	5.87%	22474
Class III-B	(37)	20	3.23%	41288	3.43%	16732	3.43%	16252
Average	(61)	42	2.95%	43803	3.01%	18875	3.02%	16102

Table 10: CPLEX B&B, 10 hours CPU Time Limit, Solved and Unsolved Problems

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 valid inequalities: 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 metaheuristic principles. Finally, we have presented computational results conducted on a large set of instances.

The computational study shows that our implementation of cover and flow cover inequalities outperforms that of the state-of-the-art software CPLEX and that the other valid inequalities contribute to further bound improvement. This conclusion points out to interesting research avenues. Although we have tested our cutting-plane algorithm within an enumerative framework, the procedure can be included in a more promising branch-and-cut algorithm, since the cuts used are valid at all nodes of the enumeration tree. An additional important advantage of the cutting-plane method we propose is that it uses valid inequalities derived from any cutset structure, which constitutes one of the most fundamental structures found in almost all network design problems. Thus, it would be interesting to investigate the usefulness of the proposed cutset generation methods to improve the formulations of other network design formulations.

Acknowledgments

While working on this project, the second author was the NSERC Industrial Research Chair on Logistics Management, ESG UQAM, and 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 experimentations.

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

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

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

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

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

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

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

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

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

[19] 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.

[20] B. Gendron and T.G. Crainic. Bounding procedures for multicommodity capacitated fixed charge network design problem. Technical Report CRT-96-06, Center for research on transportation, 1996.

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

[22] 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.

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

[24] 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.

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

[26] 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.

[27] ILOG Inc. Using the CPLEX Callable Library and CPLEX Mixed Integer Library. *CPLEX version 10.1.*, 2005.

[28] 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.

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

[30] 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.

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

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

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

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

[35] 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.

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

[37] 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.

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

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

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