EURO J Comput Optim (2014) 2:55–75
DOI 10.1007/s13675-014-0020-9

ORIGINAL PAPER

Branch-and-price-and-cut for large-scale multicommodity capacitated fixed-charge network design

Bernard Gendron · Mathieu Larose

Received: 14 November 2012 / Accepted: 13 February 2014 / Published online: 6 March 2014

Abstract We present a branch-and-price-and-cut algorithm for solving large-scale instances of the multicommodity capacitated fixed-charge network design problem. We assume good feasible solutions are already known and we focus on an efficient algorithm for proving the optimality of the solutions. The restricted master problem solved at each column generation iteration is obtained directly from the compact arc-based model by considering only a subset of the commodity flow variables. The pricing subproblem corresponds to a Lagrangian relaxation of the flow conservation and capacity constraints, leaving in the Lagrangian subproblem only the strong inequalities. The column generation procedure is completed by a cut generation step based on strong inequalities. The resulting column-and-row generation procedure is embedded within an enumerative scheme. Computational experiments on a large set of randomly generated instances are presented and analyzed.

Keywords Multicommodity capacitated fixed-charge network design · Column generation · Branch-and-price-and-cut

Mathematics Subject Classification 90C11 Mixed integer programming · 90C35 Programming involving graphs or networks · 90B10 Network models, deterministic

B. Gendron (✉) · M. Larose
Département d'informatique et de recherche opérationnelle,
Université de Montréal, Montréal, Canada
e-mail: Bernard.Gendron@cirrelt.ca

B. Gendron · M. Larose
Interuniversity Research Centre on Enterprise Networks,
Logistics and Transportation (CIRRELT), Montréal, Canada
e-mail: Mathieu.Larose@cirrelt.ca

Introduction

In this paper, we present a branch-and-price-and-cut (B&P&C) algorithm for the multicommodity capacitated fixed-charge network design problem (MCND), an NP-hard problem (Magnanti and Wong 1984) defined on a directed graph $G = (N, A)$, where N is the set of nodes and A is the set of arcs. Each commodity $k \in K$ is characterized by a demand $d^k > 0$ to be routed from an origin $O(k)$ to a destination $D(k)$. On each arc (i, j), there is a capacity $u_{ij} > 0$ on the flow of all commodities circulating on the arc (we assume $u_{ij} \leq \sum_{k \in K} d^k$). The problem is to satisfy the demands at minimum cost, while respecting the capacity constraints. 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 and fixed design costs on arc (i, j) are denoted $c_{ij} \geq 0$ and $f_{ij} \geq 0$, respectively.

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

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

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

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

$$x_{ij}^k \leq d^k y_{ij}, \quad (i, j) \in A,\ k \in K, \tag{4}$$

$$x_{ij}^k \geq 0, \quad (i, j) \in A,\ k \in K, \tag{5}$$

$$y_{ij} \in \{0, 1\}, \quad (i, j) \in A, \tag{6}$$

where $N_i(+) = \{j \in N | (i, j) \in A\}$ and $N_i(-) = \{j \in N | (j, i) \in A\}$. Equations (2) are the flow conservation constraints for each node and each commodity. The capacity constraints (3) ensure that the capacity on each arc cannot be exceeded, while forbidding any flow to circulate through an arc that is not chosen as part of the design. The so-called *strong inequalities*, (4), also serve the same purpose and are, therefore, redundant; however, they significantly improve the linear programming (LP) relaxation bounds (Crainic et al. 1999). Model (1)–(6) is the strong formulation of the MCND, the weak formulation being obtained by removing constraints (4); their corresponding LP relaxations are, respectively, the strong and the weak relaxations.

LP-based branch-and-bound (B&B) algorithms can be used to solve the MCND, but instances with a large number (typically, hundreds) of commodities remain computationally elusive. On the one hand, if strong inequalities are removed, the resulting LP relaxation provides too weak lower bounds to be used within an enumerative approach; on the other hand, if all strong inequalities are included a priori in the model, the resulting LP relaxation is not only very large, but also highly degenerate.

The obvious alternative is to add these valid inequalities in a dynamic way within a cutting-plane algorithm (Chouman et al. 2011) that might also include other well-known valid inequalities for fixed-charge network flow problems, such as cover and flow cover inequalities. This approach has been applied successfully to other, closely related, network design problems (Aardal 1988; Aardal et al. 1995; Atamtürk 2002; Atamtürk and Rajan 2002; Barahona 1996; Bienstock et al. 1998; Bienstock and Günlük 1996; Gabrel et al. 1999; Günlük 1999; Leung and Magnanti 1989; Magnanti et al. 1993, 1995; Ortega and Wolsey 2003; Raack et al. 2011). The following key observations can be derived from the computational results (Chouman et al. 2011) obtained with the cutting-plane algorithm:

- It is essential to add the strong inequalities to obtain effective lower bounds that can be computed efficiently; more precisely, cover and flow cover inequalities can be used instead of the strong inequalities to derive equally effective lower bounds, but with a much greater computational effort.
- Obviously, adding cover and flow cover inequalities to the strong relaxation improves the lower bound, but on instances with a large number of commodities (typically, hundreds of commodities), this improvement is so small that is does not payoff when the cutting-plane method is used within B&B.
- Many instances with a large number of commodities remain difficult to solve.

These findings motivate the development of our B&P&C algorithm: at each node of the enumeration tree, we solve the strong relaxation by adding in a dynamic way not only the strong inequalities, but also the flow variables. The resulting *column-and-row generation* procedure, when embedded within an enumerative scheme, is able to solve instances with a large number of commodities more efficiently than the cutting-plane method without column generation, as we will see in Sect. 3.

Note that instances with a large number of commodities typically exhibit many arcs $(i, j) \in A$ with d^k much smaller than u_{ij} for many commodities $k \in K$. Thus, a large number of commodities implies that the strong inequalities are more effective. A large number of commodities is also obviously correlated with an increase in the number of variables and constraints. Hence, as we will see in Sect. 3, the column-and-row generation method performance improves as the number of commodities increases, both in terms of its effectiveness (because strong inequalities are then more effective), but also in terms of its efficiency (because more commodities involve more variables and more constraints, which are handled efficiently by the column-and-row generation method).

Other decomposition methods have been proposed for solving the MCND, in particular Benders decomposition (Costa et al. 2012, 2009) and Lagrangian-based procedures (Crainic et al. 1999, 2001; Frangioni and Gorgone 2013; Gendron and Crainic 1994; Holmberg and Yuan 2000; Kliewer and Timajev 2005; Sellmann et al. 2002); in Sect. 2.6, we examine the relationships between our approach and some of these methods. A large number of heuristic methods have also been proposed for computing high-quality feasible solutions (Crainic et al. 2000; Crainic and Gendreau 2002; Crainic et al. 2004; Ghamlouche et al. 2003, 2004; Hewitt et al. 2010; Katayama et al. 2009; Rodríguez-Martín and Salazar-González 2010); in practice, near-optimal solutions can be obtained by these heuristics, even for large-scale instances, but the

difficulty is in assessing the quality of these solutions with effective lower bounds and in proving optimality, which is the focus of our paper.

The paper is organized as follows. Section 2 presents the B&P&C algorithm, while the results of experiments on a large set of randomly generated instances are reported in Sect. 3. We conclude this paper with a discussion of future research avenues.

Branch-and-price-and-cut algorithm

In this section, we present the different components of our B&P&C algorithm: the restricted master problem (RMP), in Sect. 2.1, the pricing subproblem, in Sect. 2.2, the cut generation, in Sect. 2.3, and the variable fixing and branching procedures, in Sect. 2.4. The overall algorithm is summarized in Sect. 2.5. We conclude this section by studying the relationships between our approach and other decomposition methods for the MCND; in particular, we show that our algorithm can be seen as a special case of two recently proposed generic B&P&C frameworks (Frangioni and Gendron 2013; Muter et al. 2012).

Restricted master problem

To obtain the RMP solved at each column-and-row generation iteration, we enlarge the set of arcs with one *artificial arc* connecting $O(k)$ to $D(k)$ for each commodity k; this arc is uncapacitated, has no fixed design cost and is given a very large transportation cost. By adding these arcs, we ensure that every RMP is always feasible; in addition, if, at any given node of the B&P&C tree, an optimal solution to the corresponding strong relaxation is obtained that includes at least one artificial arc, then the node is infeasible and therefore fathomed. We denote by A_+ the set containing A along with the artificial arcs.

To define the RMP, we associate with each arc (i, j) in A_+ a commodity subset $\widetilde{K}_{ij} \subseteq K$ over which the existing flow variables (x_{ij}^k for $k \in \widetilde{K}_{ij}$) are defined (for an artificial arc $(i, j) = (O(k), D(k))$, we always have $\widetilde{K}_{ij} = \{k\}$). In addition, we define, for each commodity k, the sets $\widetilde{A}^k = \{(i, j) \in A_+ | k \in \widetilde{K}_{ij}\}$, and for each node i and commodity k, the sets $\widetilde{N}_i^k(+) = \{j \in N | (i, j) \in \widetilde{A}^k\}$ and $\widetilde{N}_i^k(-) = \{j \in N | (j, i) \in \widetilde{A}^k\}$. Only the strong inequalities that are generated as cuts are included in the model; for each arc (i, j), we represent the generated strong inequalities using a subset $\overline{K}_{ij}$ of $\widetilde{K}_{ij}$. The RMP has thus the following form, where the dual variables are shown in parentheses on the right-hand side of each constraint:

$$\min \sum_{(i,j)\in A_+} \sum_{k\in\widetilde{K}_{ij}} c_{ij} x_{ij}^k + \sum_{(i,j)\in A_+} f_{ij} y_{ij}, \tag{7}$$

$$\sum_{j\in\widetilde{N}_i^k(+)} x_{ij}^k - \sum_{j\in\widetilde{N}_i^k(-)} x_{ji}^k = \begin{cases} d^k, & \text{if } i = O(k), \\ -d^k, & \text{if } i = D(k), \\ 0, & \text{otherwise,} \end{cases} \quad i \in N,\ k \in K, \quad (\pi_i^k) \tag{8}$$

$$\sum_{k\in\widetilde{K}_{ij}} x_{ij}^k \le u_{ij} y_{ij}, \quad (i, j) \in A_+, \quad (\alpha_{ij}) \tag{9}$$

$$x_{ij}^k \le d^k y_{ij}, \quad (i,j) \in A_+,\ k \in \overline{K}_{ij} \subseteq \widetilde{K}_{ij}, \quad (\beta_{ij}^k) \tag{10}$$

$$y_{ij} \le 1, \quad (i,j) \in A_+, \quad (\gamma_{ij}) \tag{11}$$

$$x_{ij}^k \ge 0, \quad (i,j) \in A_+, \quad k \in \widetilde{K}_{ij}, \tag{12}$$

$$y_{ij} \ge 0, \quad (i,j) \in A_+. \tag{13}$$

At the root node of the B&P&C tree, only the variables associated with the artificial arcs are handled by the initial RMP. After solving the RMP at each column generation iteration, flow variables with negative reduced cost are added (see Sect. 2.2) and the new RMP is solved. This column generation process ends when there are no more flow variables with negative reduced cost. At this point, the cut generation step is performed (see Sect. 2.3). If cuts are added, the column generation process restarts, with an initial solution derived by the dual simplex method. At subsequent nodes of the B&P&C tree, an initial solution is also derived by the dual simplex method, considering the addition of the branching constraint (see Sect. 2.4) to the currently available basic solution. For further details on the development and implementation of column generation and B&P&C algorithms, the reader is referred to the abundant literature on the topic (see for instance Desaulniers et al. 2005; Lübbecke and Desrosiers 2005 and the references therein).

Pricing subproblem

To add flow variables with negative reduced cost to the RMP, we need to seek arcs $(i,j) \in A$ and commodities $k \notin \widetilde{K}_{ij}$ such that $c_{ij} - \pi_i^k + \pi_j^k + \alpha_{ij} + \beta_{ij}^k < 0$. We note that the values of the dual variables are known after solving the RMP, with the exception of the β_{ij}^k associated with the strong inequalities (10). One simple way of solving this issue is to simply forget about them and generate all flow variables corresponding to arcs $(i,j) \in A$ and commodities $k \notin \widetilde{K}_{ij}$ such that $c_{ij} - \pi_i^k + \pi_j^k + \alpha_{ij} < 0$. The resulting column generation method would still be valid, but after every pricing iteration, potentially many flow variables with nonnegative reduced cost could be generated, i.e., those associated with $(i,j) \in A$ and $k \notin \widetilde{K}_{ij}$ such that $c_{ij} - \pi_i^k + \pi_j^k + \alpha_{ij} < 0$ and $\beta_{ij}^k \ge -(c_{ij} - \pi_i^k + \pi_j^k + \alpha_{ij})$. A more efficient approach consists in computing the values of the β_{ij}^k variables to add to the RMP only flow variables with negative reduced cost.

To achieve this objective, we first write down the dual and the complementary slackness conditions of the LP relaxation of model (1)–(6):

$$\max \sum_{k \in K} d^k \left(\pi_{O(k)}^k - \pi_{D(k)}^k \right) - \sum_{(i,j) \in A} \gamma_{ij} \tag{14}$$

$$\pi_i^k - \pi_j^k - \alpha_{ij} - \beta_{ij}^k \le c_{ij}, \quad (i,j) \in A,\ k \in K, \quad (x_{ij}^k) \tag{15}$$

$$u_{ij}\alpha_{ij} + \sum_{k \in K} d^k \beta_{ij}^k - \gamma_{ij} \le f_{ij}, \quad (i,j) \in A, \quad (y_{ij}) \tag{16}$$

$$\alpha_{ij} \ge 0, \quad (i,j) \in A, \tag{17}$$

$$\beta_{ij}^k \ge 0, \quad (i,j) \in A,\ k \in K, \tag{18}$$

$$\gamma_{ij} \ge 0, \quad (i,j) \in A, \tag{19}$$

$$x_{ij}^k \left(c_{ij} - \pi_i^k + \pi_j^k + \alpha_{ij} + \beta_{ij}^k \right) = 0, \quad (i, j) \in A,\ k \in K \tag{20}$$

$$y_{ij} \left(f_{ij} + \gamma_{ij} - u_{ij}\alpha_{ij} - \sum_{k \in K} d^k \beta_{ij}^k \right) = 0, \quad (i, j) \in A, \tag{21}$$

$$\alpha_{ij} \left(u_{ij} y_{ij} - \sum_{k \in K} x_{ij}^k \right) = 0, \quad (i, j) \in A, \tag{22}$$

$$\beta_{ij}^k \left(d^k y_{ij} - x_{ij}^k \right) = 0, \quad (i, j) \in A,\ k \in K, \tag{23}$$

$$\gamma_{ij} \left(1 - y_{ij} \right) = 0, \quad (i, j) \in A. \tag{24}$$

In the remainder of the paper, we use the following notation: for any $(i, j) \in A$, we define $c_{ij}^k(\pi, \alpha) \equiv c_{ij} - \pi_i^k + \pi_j^k + \alpha_{ij}$, for any $k \in K$, and $f_{ij}(\alpha) \equiv f_{ij} - u_{ij}\alpha_{ij}$. The analysis is based on deriving reduced cost optimality conditions from (14)–(24), for any arc $(i, j) \in A$ and for all commodities $k \in K$. If $k \in \widetilde{K}_{ij}$, these conditions are automatically satisfied, since the RMP is solved to optimality. Note that, for $k \in \widetilde{K}_{ij} \backslash \overline{K}_{ij}$, the corresponding strong inequality is not in the RMP, which implies that $\beta_{ij}^k \equiv 0$, but the corresponding flow variable x_{ij}^k is already in the RMP, so we do not need to consider this case. The goal is to add to the RMP only the flow variables that correspond to the commodities $k \notin \widetilde{K}_{ij}$ that do not satisfy the reduced cost optimality conditions. We denote by $\overline{y}, \overline{x}$ and $\overline{\pi}, \overline{\alpha}, \overline{\beta}, \overline{\gamma}$, respectively, the optimal solutions to the RMP and to its dual; the optimal solution to the RMP is completed in the obvious way by setting $\overline{x}_{ij}^k = 0$, for any $(i, j) \in A$ and all $k \notin \widetilde{K}_{ij}$. For any arc (i, j), we distinguish two cases:

- $\overline{y}_{ij} > 0$. Let $k \notin \widetilde{K}_{ij}$; for the solution to the RMP to be optimal for the LP relaxation of model (1)–(6), we must have, by complementary slackness condition (23)

$$\overline{\beta}_{ij}^k (\underbrace{d^k \overline{y}_{ij}}_{>0} - \underbrace{\overline{x}_{ij}^k}_{=0}) = 0 \Longrightarrow \overline{\beta}_{ij}^k = 0.$$

  The reduced cost optimality condition for $k \notin \widetilde{K}_{ij}$ is therefore $c_{ij}^k(\overline{\pi}, \overline{\alpha}) \geq 0$, which implies that we add to the RMP the flow variables associated with arc (i, j) and commodity $k \notin \widetilde{K}_{ij}$ such that $c_{ij}^k(\overline{\pi}, \overline{\alpha}) < 0$.
- $\overline{y}_{ij} = 0$. In this case, we have $\overline{x}_{ij}^k = 0$ for all $k \in K$ and

$$\overline{\gamma}_{ij} (\underbrace{1 - \overline{y}_{ij}}_{\neq 0}) = 0 \Longrightarrow \overline{\gamma}_{ij} = 0.$$

  For the solution to the RMP to be optimal for the LP relaxation of model (1)–(6), we, therefore, must have, by the dual constraints (16), the following reduced cost optimality conditions:

$$f_{ij}(\overline{\alpha}) \geq \sum_{k \in K} d^k \overline{\beta}_{ij}^k.$$

By the dual constraints (15) and (18), we must have $\overline{\beta}_{ij}^k \geq \max\{0, -c_{ij}^k(\overline{\pi}, \overline{\alpha})\}$ for any $k \in K$. This implies that we add to the RMP the flow variables associated with arc (i, j) such that

$$f_{ij}(\overline{\alpha}) < \sum_{k \in K} d^k \max\{0, -c_{ij}^k(\overline{\pi}, \overline{\alpha})\},$$

but only for commodities $k \notin \widetilde{K}_{ij}$ such that $c_{ij}^k(\overline{\pi}, \overline{\alpha}) < 0$.

To summarize, the pricing subproblem consists in performing the following tests for all arcs (i, j):

1. If $\overline{y}_{ij} > 0$, then for any $k \notin \widetilde{K}_{ij}$ such that $c_{ij}^k(\overline{\pi}, \overline{\alpha}) < 0$, we add the flow variables x_{ij}^k to the RMP.
2. If $\overline{y}_{ij} = 0$ and $f_{ij}(\overline{\alpha}) < \sum_{k \in K} d^k \max\{0, -c_{ij}^k(\overline{\pi}, \overline{\alpha})\}$, then for any $k \notin \widetilde{K}_{ij}$ such that $c_{ij}^k(\overline{\pi}, \overline{\alpha}) < 0$, we add the flow variables x_{ij}^k to the RMP.

Cut generation

To solve the RMP at each column generation iteration, one option is to add *a priori* all the strong inequalities (10). However, as mentioned in the "Introduction", it is more efficient to add them in a dynamic way. In particular, the separation problem is trivial: for each arc (i, j), we generate the strong inequalities for all commodities $k \in \widetilde{K}_{ij}$ such that $\overline{x}_{ij}^k > d^k \overline{y}_{ij}$. The addition of the corresponding constraints makes the current primal basic solution infeasible, so the dual simplex method is performed until a feasible primal solution is obtained. The column generation procedure is then restarted. Cut generation is performed after the column generation procedure, when no more flow variables with negative reduced cost can be found.

Variable fixing and branching

At any node of the B&P&C algorithm, we apply reduced cost variable fixing at each iteration of the cut generation procedure. The technique is well known (Savelsbergh 1994): given the current LP relaxation lower bound Z^l and the best-known upper bound Z^*, we test, for each arc (i, j), if $Z^l + |\overline{f}_{ij}| \geq Z^*$, in which case we can fix y_{ij} to $\overline{y}_{ij}$, where $\overline{f}_{ij} = f_{ij} - u_{ij}\overline{\alpha}_{ij} - \sum_{k \in K} d^k \overline{\beta}_{ij}^k$.

To show the validity of this test, first note that if $0 < \overline{y}_{ij} < 1$, then $\overline{f}_{ij} = 0$ by complementary slackness conditions (21) and (24), which implies that $Z^l + |\overline{f}_{ij}| = Z^l < Z^*$; otherwise, the current node could be fathomed. Then, there are two remaining cases:

- $\overline{y}_{ij} = 0$. By complementary slackness condition (24), we have $\overline{\gamma}_{ij} = 0$. Assume we add the constraint $y_{ij} \geq 1$ with which we associate a dual variable $\lambda_{ij} \geq 0$ that is added to the objective $(+\lambda_{ij})$. The dual constraint (16) is then rewritten as $\lambda_{ij} \leq f_{ij} - u_{ij}\alpha_{ij} - \sum_{k \in K} d^k \beta_{ij}^k$. A dual feasible solution is obtained by setting

$\lambda_{ij} = \overline{f}_{ij}$, which implies that $Z^l + \overline{f}_{ij}$ is a lower bound on the problem obtained by adding the constraint $y_{ij} \geq 1$. Therefore, if $Z^l + \overline{f}_{ij} = Z^l + |\overline{f}_{ij}| \geq Z^*$, we necessarily have $y_{ij} < 1$, i.e., $y_{ij} = 0$.

- $\overline{y}_{ij} = 1$. By the objective of the dual (14) and the dual constraints (16) and (19), we have $\overline{\gamma}_{ij} = \max\{0, -\overline{f}_{ij}\}$. Assume we add the constraint $y_{ij} \leq 0$ with which we associate a dual variable $\mu_{ij} \geq 0$. The dual constraint (16) is then rewritten as $-\mu_{ij} - \gamma_{ij} \leq f_{ij} - u_{ij}\alpha_{ij} - \sum_{k \in K} d^k \beta_{ij}^k$. A dual feasible solution is obtained by setting $\mu_{ij} = -\overline{f}_{ij}$ and $\gamma_{ij} = 0$, which implies that $Z^l - \overline{f}_{ij}$ is a lower bound on the problem obtained by adding the constraint $y_{ij} \leq 0$. Therefore, if $Z^l - \overline{f}_{ij} = Z^l + |\overline{f}_{ij}| \geq Z^*$, we necessarily have $y_{ij} > 0$, i.e., $y_{ij} = 1$.

The column-and-row generation procedure is performed until the node is fathomed or no more cuts can be generated. In the latter case, branching is performed. At each node, the LP relaxation of model (1)–(6), with the addition of the variable fixing and branching constraints, is solved by the column-and-row generation procedure. Branching occurs only when $Z^l < Z^*$, a condition that implies that the LP optimal solution is fractional, i.e., there exists at least one arc (i, j) such that $0 < \overline{y}_{ij} < 1$ (otherwise, the LP optimal solution is feasible and if its value, Z^l, is less than Z^*, it replaces it as the best incumbent value). In this case, we select one such arc and create the two nodes with the added branching constraints $y_{ij} = 0$ and $y_{ij} = 1$. To select the branching variable, we use *reliability branching* (as implemented in the SCIP library Achterberg 2009), certainly one of the most efficient branching rule available in general-purpose LP-based B&B MIP solvers (our preliminary tests have confirmed this assessment). Reliability branching is similar to the classical pseudocost branching rule, except that a candidate variable (with a fractional value in the LP relaxation) is declared "unreliable" if the number of computations of objective gains obtained by branching on this variable is "too small," as measured by the so-called reliability parameter. When a variable is "unreliable," the gains in the objective when branching on this variable are computed by solving approximately the resulting LP relaxations with a limited number of dual simplex iterations, as in the strong branching rule. For more details on LP-based B&B branching rules, see (Achterberg et al. 2005).

Summary of the algorithm

The algorithm is outlined as follows (the steps are commented below):

1. Initialize the upper bound Z^* and the incumbent solution.
2. Initialize the node pool $\mathcal{L}$ with the root node.
3. *Selection* Select the next node to evaluate in $\mathcal{L}$ and remove it from $\mathcal{L}$.
4. *Lower bound* Solve the LP relaxation at the current node:
   (a) Find a pair of primal-dual solutions $\overline{y}, \overline{x}$ and $\overline{\pi}, \overline{\alpha}$ by solving the RMP with the dual simplex method.
   Column generation:
   (b) *Pricing* For each arc (i, j), if $\overline{y}_{ij} > 0$ or ($\overline{y}_{ij} = 0$ and $f_{ij}(\overline{\alpha}) < \sum_{k \in K} d^k \max\{0, -c_{ij}^k(\overline{\pi}, \overline{\alpha})\}$), then for any $k \notin \widetilde{K}_{ij}$ such that $c_{ij}^k(\overline{\pi}, \overline{\alpha}) < 0$, add the flow variables x_{ij}^k to the RMP.

(c) If flow variables were added to the RMP, solve the new RMP by the primal simplex method to obtain a pair of primal-dual solutions $\overline{y}, \overline{x}$ and $\overline{\pi}, \overline{\alpha}$; go to step 4b.
(d) Let Z^l be the lower bound obtained after the column generation procedure; if $\overline{y}$ is integer and $Z^l < Z^*$, then let $Z^* = Z^l$ and store $\overline{y}, \overline{x}$ as the new incumbent solution.
(e) If $Z^l \geq Z^*$, then go to step 6.
Cut generation:
(f) *Separation* For each arc (i, j) and for any $k \in \widetilde{K}_{ij}$ such that $\overline{x}^k_{ij} > d^k \overline{y}_{ij}$, add the corresponding strong inequalities to the RMP.
(g) If strong inequalities were added to the RMP:
  i. Solve the new RMP by the dual simplex method to obtain a solution $\overline{y}, \overline{x}$ and a lower bound Z^l.
  ii. If $\overline{y}$ is integer and $Z^l < Z^*$, then let $Z^* = Z^l$ and store $\overline{y}, \overline{x}$ as the new incumbent solution.
  iii. If $Z^l \geq Z^*$, then go to step 6.
  iv. *Variable fixing:* For each arc (i, j), if $Z^l + |\overline{f}_{ij}| \geq Z^*$, then fix y_{ij} to $\overline{y}_{ij}$; if some variables were fixed, go to step 4a.
  v. Go to step 4b.

5. *Branching* If $Z^l < Z^*$, perform branching to generate two child nodes inserted in $\mathcal{L}$.
6. If $\mathcal{L} = \emptyset$, stop the algorithm; otherwise, go to step 3.

In step 1, a feasible solution and an upper bound is obtained by any heuristic method to solve the MCND. As mentioned in the Introduction, there are several effective heuristic approaches that identify near-optimal solutions, even for large-scale instances. In step 2, the pool of B&P&C nodes, noted $\mathcal{L}$, is initialized with the root node. In step 3, the next node to evaluate is selected and removed from $\mathcal{L}$. From our computational results, we recommend the best-first selection rule (see Sect. 3; for more details on B&B node selection rules, see for instance Achterberg 2009; Atamtürk and Savelsbergh 2005; Ibaraki 1988). Step 4 contains the details of the column-and-row generation procedure used at each node. The goal of step 4a is to find an initial pair of primal-dual solutions to start the column generation procedure. At the very beginning, artificial arcs are added and flow is sent along these arcs, while subsequently, the dual simplex method is performed to take into account the effect of the additional constraints derived by variable fixing and branching. Step 4b implements the solution of the pricing subproblem to add flow variables with negative reduced cost to the RMP. In case new flow variables have been added to the RMP, the next column generation iteration proceeds; otherwise, the column generation procedure has converged, which allows to identify a lower bound Z^l. Steps 4d and 4e are the incumbent update and the lower bound test, respectively, to be performed after the column generation procedure. The separation problem for strong inequalities is then solved in step 4f. If strong inequalities have been generated, the RMP with the added cuts is solved by the dual simplex method in step 4(g)i, followed by the incumbent update and the lower bound test in steps 4(g)ii and 4(g)iii, then by variable fixing in step 4(g)iv, after which the control is transferred to the column generation procedure. In case no more strong

inequalities have been generated, the column-and-row generation procedure stops. Finally, step 5 performs the branching operation and step 6 verifies the termination of the B&P&C algorithm, i.e., the node pool $\mathcal{L}$ is empty; we can also stop the algorithm when a time limit has been reached.

Relationships to other decomposition methods

In this section, we discuss the relationships between our B&P&C algorithm for the MCND and other decomposition methods, both specific to the MCND and general ones. We first consider the Lagrangian relaxation with respect to the flow conservation equations (2) and the capacity constraints (3), where we associate with each set of constraints, Lagrange multipliers π and $\alpha \geq 0$, respectively. To the best of our knowledge, this particular Lagrangian relaxation has not been studied in the literature on the MCND. The corresponding Lagrangian subproblem decomposes by arc and can be written as follows, for each arc (i, j) (recall that $c^k_{ij}(\pi, \alpha) \equiv c_{ij} - \pi^k_i + \pi^k_j + \alpha_{ij}$, for any $k \in K$, and $f_{ij}(\alpha) \equiv f_{ij} - u_{ij}\alpha_{ij}$):

$$Z_{ij}(\pi, \alpha) = \min \sum_{k \in K} c^k_{ij}(\pi, \alpha) x^k_{ij} + f_{ij}(\alpha) y_{ij} \tag{25}$$

$$0 \leq x^k_{ij} \leq d^k y_{ij}, \quad k \in K, \tag{26}$$

$$y_{ij} \in \{0, 1\}. \tag{27}$$

The associated Lagrangian dual is

$$\max_{\pi, \alpha \geq 0} \sum_{k \in K} d^k \left(\pi^k_{O(k)} - \pi^k_{D(k)} \right) + \sum_{(i,j) \in A} Z_{ij}(\pi, \alpha). \tag{28}$$

To solve the Lagrangian subproblem, we consider the two possibilities, $y_{ij} = 0$ and $y_{ij} = 1$. If $y_{ij} = 0$, the only feasible solution is to set all flow variables to 0, with an objective function value equal to 0. If $y_{ij} = 1$, an optimal solution is obtained by setting $x^k_{ij} = d^k$, if $c^k_{ij}(\pi, \alpha) < 0$, and $x^k_{ij} = 0$, otherwise; the corresponding objective function value is $\sum_{k \in K} - \max\{0, -c^k_{ij}(\pi, \alpha)\} d^k + f_{ij}(\alpha)$. An optimal solution is obtained by selecting the alternative with the smallest objective function value; thus, the structure of an optimal solution $\widetilde{y}, \widetilde{x}$ to the Lagrangian subproblem for arc (i, j) is

$$\widetilde{y}_{ij} = \begin{cases} 1, & \text{if } \sum_{k \in K} - \max\{0, -c^k_{ij}(\pi, \alpha)\} < 0, \\ 0, & \text{if } \sum_{k \in K} - \max\{0, -c^k_{ij}(\pi, \alpha)\} > 0, \\ 0 \text{ or } 1, & \text{otherwise,} \end{cases} \tag{29}$$

$$\widetilde{x}^k_{ij} = \begin{cases} d^k, & \text{if } \widetilde{y}_{ij} = 1 \text{ and } c^k_{ij}(\pi, \alpha) < 0, \\ 0, & \text{if } \widetilde{y}_{ij} = 0 \text{ or } (\widetilde{y}_{ij} = 1 \text{ and } c^k_{ij}(\pi, \alpha) > 0), \\ 0 \text{ or } d^k, & \text{otherwise,} \end{cases} \quad k \in K. \tag{30}$$

Proposition 1 *The solution to the pricing subproblem presented in Sect. 2.2 corresponds to one of these optimal solutions for $\pi = \overline{\pi}$ and $\alpha = \overline{\alpha}$, in the sense that a flow variable x^k_{ij} is added to the RMP when this optimal solution satisfies $\widetilde{x}^k_{ij} = d^k$.*

Proof Consider first the case $\overline{y}_{ij} > 0$. We then have, for any $k \in K$, $\overline{\beta}^k_{ij} = -c^k_{ij}(\pi, \alpha) \geq 0$ if $\overline{x}^k_{ij} > 0$, or $\overline{\beta}^k_{ij} = 0$ if $\overline{x}^k_{ij} = 0$, by complementary slackness conditions (20) and (23); this implies that $\overline{\beta}^k_{ij} = \max\{0, -c^k_{ij}(\pi, \alpha)\}$. By complementary slackness condition (21), it follows that

$$\begin{aligned} f_{ij}(\overline{\alpha}) &= \sum_{k \in K} d^k \overline{\beta}^k_{ij} - \overline{\gamma}_{ij} \\ &= \sum_{k \in K} d^k \max\{0, -c^k_{ij}(\pi, \alpha)\} - \overline{\gamma}_{ij} \\ &\leq \sum_{k \in K} d^k \max\{0, -c^k_{ij}(\pi, \alpha)\}. \end{aligned}$$

Therefore, when $\overline{y}_{ij} > 0$, an optimal solution to the corresponding Lagrangian subproblem with $\pi = \overline{\pi}$ and $\alpha = \overline{\alpha}$ is $\widetilde{y}_{ij} = 1$ and, for any $k \in K$, $\widetilde{x}^k_{ij} = d^k$, if $c^k_{ij}(\pi, \alpha) < 0$, and 0, otherwise.

Now, consider the case $\overline{y}_{ij} = 0$; an optimal solution to the corresponding Lagrangian subproblem with $\pi = \overline{\pi}$ and $\alpha = \overline{\alpha}$ is 1) $\widetilde{y}_{ij} = 1$, if $f_{ij}(\overline{\alpha}) < \sum_{k \in K} d^k \max\{0, -c^k_{ij}(\pi, \alpha)\}$ and, for any $k \in K$, $\widetilde{x}^k_{ij} = d^k$, if $c^k_{ij}(\pi, \alpha) < 0$, and 0, otherwise; or 2) $\widetilde{y}_{ij} = 0$, if $f_{ij}(\overline{\alpha}) \geq \sum_{k \in K} d^k \max\{0, -c^k_{ij}(\pi, \alpha)\}$ and $\widetilde{x}^k_{ij} = 0$, for all $k \in K$.

In both cases, we conclude that the pricing subproblem is equivalent to solving the Lagrangian subproblem with $\pi = \overline{\pi}$ and $\alpha = \overline{\alpha}$. □

Thus, the column-and-row generation procedure can be interpreted as a solution method for the Lagrangian dual (28), where the Lagrange multipliers are obtained by solving the restricted master problem (7)–(13). While this approach is similar to Dantzig–Wolfe (DW) decomposition, it is fundamentally different, in that DW decomposition is based on a reformulation of the problem in terms of the (*exponentially many*) extreme points of the convex hull of the feasible solution set to the Lagrangian subproblem. Our column-and-row generation procedure is in fact a special case of the *structured DW (SDW)* decomposition framework presented in Frangioni and Gendron (2013), which encompasses DW decomposition by allowing other reformulations than the standard DW one. The network design example used to illustrate the SDW decomposition in Frangioni and Gendron (2013) is based on reformulating general integer variables with binary variables, thus increasing the size of the variable space from a polynomial number to a pseudo-polynomial number. In contrast, our application of SDW to the MCND is working on the same formulation as the original one, i.e., the "reformulation" is the original formulation itself, which has a polynomial number of variables.

Our column-and-row generation procedure can also be seen as a special case of the framework presented in Muter et al. (2012) for solving large-scale LPs with so-called *column-dependent-rows*, which are linking constraints (between two types of variables) that are "too many" to be included in the model directly or that can only be known when all variables are explicitly generated. Clearly, the strong inequalities (4)

correspond to the first category of column-dependent-rows, since they are linking the flow and the design variables, and they are "too many", although not exponentially many, as in most applications presented in Muter et al. (2012).

The Lagrangian interpretation of our column-and-row generation procedure allows us to clarify its relationships to existing Lagrangian methods for the MCND. These methods are based on two different Lagrangian relaxations: the so-called *shortest path* (or flow) and *knapsack* relaxations. The first one relaxes both the capacity constraints (3) and the strong inequalities (4), leading to a Lagrangian subproblem that decomposes into a set of shortest path problems for each commodity and a problem in y variables solvable by inspection (Crainic et al. 1999, 2001; Frangioni and Gorgone 2013; Gendron and Crainic 1994). The second Lagrangian approach relaxes the flow conservation equations (2); the corresponding Lagrangian subproblem decomposes by arc and, for each arc, it consists in solving a continuous knapsack problem (Crainic et al. 1999, 2001; Gendron and Crainic 1994; Holmberg and Yuan 2000; Kliewer and Timajev 2005; Sellmann et al. 2002). In all these references, the Lagrangian dual is solved either by a subgradient approach or by a bundle method, which is a generalized form of DW decomposition, where the master problem is "stabilized" with the addition of a convex (usually quadratic) term to the objective function (Frangioni 2005). Thus, our column-and-row generation procedure differs from the existing Lagrangian methods in two aspects: first, both the flow conservation equations (2) and the capacity constraints (3) are relaxed, so the resulting Lagrangian subproblem differs significantly from the shortest path relaxation, while it is similar to the knapsack relaxation, but simpler, since it does not contain the capacity constraints; second, the Lagrangian dual is solved by the column-and-row generation procedure, not by subgradient or bundle methods. It is noteworthy that the master problem used in bundle methods can be seen as a generalized form of the standard DW reformulation, with exponentially many variables, while the master problem in our approach is nothing but the original, compact, formulation.

Column(-and-row) generation methods for multicommodity flow problems are often identified with the classical path-based model, where flow variables are represented by paths between each O-D pair. In fact, for the MCND, the DW master problem associated with the shortest path relaxation corresponds to the path-based model; thus, solving the associated Lagrangian dual by a bundle method implicitly makes use of the path-based formulation. The path-based model has also been used for solving the multicommodity flow subproblems derived from fixing the design variables (Kliewer and Timajev 2005; Sellmann et al. 2002), but also in sophisticated heuristic methods (Crainic et al. 2000; Katayama et al. 2009). In particular, the column-and-row generation procedure embedded in the capacity scaling approach of Katayama et al. (2009) is similar to ours, in that it uses a cutting-plane procedure to generate strong inequalities, but is also different, since the flow variables are represented by paths, not by arcs as in our approach.

In this Section, we have highlighted three interesting features of our method:

- It can be seen as a Lagrangian relaxation method, where the Lagrangian subproblem results from relaxing the flow conservation and the capacity constraints, a Lagrangian relaxation that has not been studied before.

- It is a special (simple) case of structured Dantzig-Wolfe decomposition, a framework recently presented in Frangioni and Gendron (2013).
- It is different from the other column generation methods proposed for similar problems, which use path-based models and can be seen as standard Dantzig-Wolfe decomposition applied to the shortest path relaxation.

Computational results

This section presents computational results obtained by the B&P&C algorithm on a publicly available set of 196 instances (the so-called "Canad" instances, see Frangioni 2014) used in several papers on the MCND (for instance) and described in detail in Crainic et al. (2001). 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 transportation cost and the fixed design cost. These instances are characterized by various degrees of capacity tightness, with regard to the total demand, and importance of the fixed design cost, with respect to the transportation cost.

The instances are divided into three classes. Class I (the "C" instances in Frangioni 2014) consists of 31 problem instances with many commodities compared to the number of nodes, while Class II (the "C+" instances in Frangioni 2014) contains 12 problem instances with few commodities compared to the number of nodes. Class III (the "R" instances in Frangioni 2014) 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 capacity tightness and importance of the fixed design cost. 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. Table 1 gives the size of the instances in each class.

Table 1 Classes and problem dimensions (number of instances in parentheses)

Class I $\lvert N\rvert, \lvert A\rvert, \lvert K\rvert$	(31)	Class II $\lvert N\rvert, \lvert A\rvert, \lvert K\rvert$	(12)	Class III-A $\lvert N\rvert, \lvert A\rvert, \lvert K\rvert$	(72)	Class III-B $\lvert N\rvert, \lvert A\rvert, \lvert K\rvert$	(81)
20, 230, 40	(3)	25, 100, 10	(3)	10, 35, 10	(6)	20, 120, 40	(9)
20, 230, 200	(4)	25, 100, 30	(3)	10, 35, 25	(6)	20, 120, 100	(9)
20, 300, 40	(4)	100, 400, 10	(3)	10, 35, 50	(6)	20, 120, 200	(9)
20, 300, 200	(4)	100, 400, 30	(3)	10, 60, 10	(9)	20, 220, 40	(9)
30, 520, 100	(4)			10, 60, 25	(9)	20, 220, 100	(9)
30, 520, 400	(4)			10, 60, 50	(9)	20, 220, 200	(9)
30, 700, 100	(4)			10, 85, 10	(9)	20, 320, 40	(9)
30, 700, 400	(4)			10, 85, 25	(9)	20, 320, 100	(9)
				10,85,50	(9)	20, 320, 200	(9)

The B&P&C method is compared against two competitors: a state-of-the-art MIP solver, CPLEX (version 12.3), and the B&C algorithm obtained by performing, at each node of the B&C tree, a cutting-plane procedure that includes only the strong inequalities as cuts. Each competing method is tested to verify two hypotheses: 1) the B&P&C algorithm is competitive with a state-of-the-art MIP solver; 2) the column-and-row generation procedure is at least as efficient as the cutting-plane procedure, and more efficient on large-scale instances with many commodities (in the order of 100). The initial model given to CPLEX is the weak formulation, i.e., model (1)–(6) without the strong inequalities (4), since including these inequalities in advance would give very poor performance. CPLEX then generates its own cuts. Both the B&C and the B&P&C algorithms are implemented in C++ using the SCIP library, version 2.0.2 (Achterberg 2009); we use the B&B implementation of SCIP without heuristics and separators, with CPLEX (version 12.3) used as the LP solver. In both B&C and B&P&C, we use reliability branching, as implemented in SCIP, while CPLEX uses its own default branching rule. All experiments are performed on an Intel Xeon X5660 operating at 2.80 GHz under the Linux operating system.

The goal of our experiments is to evaluate the performance of the B&P&C algorithm in providing effective lower bounds; in particular, we wish to evaluate the efficiency of the algorithm for proving the optimality of an already known optimal solution. Thus, for both the B&P&C method and its competitors, we give as initial incumbent the best known feasible solution for each problem instance, which is an optimal one for most of them. When performing CPLEX, we consequently deactivate all features related to the computation of feasible solutions. For all methods, we fix a "reasonable" CPU time limit of 3 h.

We first look at the results obtained at the root node. CPLEX delivers better lower bounds than the strong relaxation bound computed by B&C and B&P&C, because CPLEX can generate a large number of cuts of different types. It is interesting to assess the quality of the strong relaxation for various types of instances; thus, for each problem instance, we measure the gap between the strong relaxation bound and CPLEX lower bound. We are also interested in assessing the efficiency of the three bounding methods, the cutting-plane procedure in CPLEX, the cutting-plane approach that generates only the strong inequalities and our column-and-row generation method; we therefore measure the CPU times obtained by the three methods. The average results for each class of problem instances are presented in Table 2; performance measures are computed for each instance and their arithmetic averages are computed over all instances in a given class. Column "Class" corresponds to the class of problem instances; column "Method" relates to the three methods, CPLEX, B&C and B&P&C; column "Gap" provides the average relative gap between the strong relaxation bound and CPLEX lower bound, which is the best of the two for all problem instances (the relative gap is the difference between the two bounds divided by CPLEX lower bound); column "CPU" gives the average CPU time in seconds for each method; finally, column "CPU gain" displays the average relative gain in CPU time obtained by B&C and B&P&C over CPLEX (the relative gain in CPU time is the difference between the two CPU times divided by CPLEX CPU time).

These results show that the strong relaxation bound is very close (within 1% on average) to the lower bound computed by CPLEX for problem instances in Classes

Table 2 Comparison of the methods at the root node

Class	Method	Gap (%)	CPU (s)	CPU gain (%)
I	CPLEX	0	26.2	0
	B&C	0.2	23.3	11
	B&P&C	0.2	6.3	76
II	CPLEX	0	2.3	0
	B&C	2.5	0.6	74
	B&P&C	2.5	0.6	74
III-A	CPLEX	0	0.09	0
	B&C	0.9	0.02	77
	B&P&C	0.9	0.02	77
III-B	CPLEX	0	6.3	0
	B&C	0.2	5.4	14
	B&P&C	0.2	2.6	59

I, III-A and III-B. For Class II, the average gap between the strong relaxation bound and CPLEX attains 2.5%. This is consistent with the results reported in Chouman et al. (2011), who show that other inequalities than the strong ones, namely cover and flow cover inequalities, are really useful only for instances in Class II, providing only minor improvements for instances in the other classes. Note that, while Class II instances have few commodities (no more than 30), most instances in Classes I and III-B have many commodities (at least 100). For these two classes, the strong relaxation bound and CPLEX lower bound are almost equal, with an average gap of 0.2%; the CPU times, however, are significantly different: while the average relative gain for B&C is relatively modest (slightly more than 10%), it reaches more than 60% for B&P&C. These results are promising for the B&P&C algorithm, since they suggest significant gains in efficiency for instances in Classes I and III-B, especially for large-scale instances with hundreds of commodities. We verify this hypothesis in the remainder of this section.

To analyze the performance of the different approaches when branching is performed and the enumeration trees are explored, we examine the impact of two classical selection rules, best-first and depth-first. Although there are other B&B node selection rules (Achterberg 2009; Atamtürk and Savelsbergh 2005), these alternative rules are mostly designed to balance the effort between finding good feasible solutions and improving the lower bounds. Since we assume we start the enumeration with a very good (optimal on many instances) solution, we focus only on the two classical selection rules. In general, for any B&B algorithm, the best-first search selection rule provides reliable results, independently of the problem instance being tested, since it displays two interesting features: (1) when the algorithm proves the optimality of the solution within the time limit, the total number of nodes is generally smaller than with other selection rules (Ibaraki 1988); (2) otherwise, if the algorithm is stopped prematurely without proving the optimality of the solution, the lower bound is generally tighter than with other selection rules, since best-first search focuses the search on improving the lower bound (Atamtürk and Savelsbergh 2005). The first advantage

Table 3 Results on "Easy" instances (number of instances in parentheses)

Class	Method	CPU (s)	Nodes	Variables
I (12)	CPLEX-depth	845	1,487	28,942
	CPLEX-best	1,204	1,487	28,942
	B&C-depth	1,017	1,657	29,031
	B&C-best	1,013	1,756	29,031
	B&P&C-depth	474	2,352	6,692
	B&P&C-best	419	2,062	6,662
II (8)	CPLEX-depth	26	630	2676
	CPLEX-best	30	627	2,676
	B&C-depth	327	6,080	2,693
	B&C-best	322	5,767	2,693
	B&P&C-depth	687	7,966	1,573
	B&P&C-best	496	5,870	1,557
III-A (72)	CPLEX-depth	2	213	1,822
	CPLEX-best	2	215	1,822
	B&C-depth	2	285	1,850
	B&C-best	3	303	1,850
	B&P&C-depth	2	421	745
	B&P&C-best	3	396	743
III-B (60)	CPLEX-depth	766	3,516	19,409
	CPLEX-best	1,051	3,507	19,409
	B&C-depth	1,114	1,5874	19,510
	B&C-best	1,213	15,926	19,510
	B&P&C-depth	901	21,131	7,739
	B&P&C-best	855	16,829	7,733

vanishes, however, under two conditions: (a) the initial incumbent is an optimal solution; (b) branching and bounding operations provide the same output, irrespective of the selection rule; indeed, under conditions (a) and (b), the total number of nodes is the same, independently of the selection rule (Ibaraki 1988). Thus, when the initial incumbent is close to an optimal solution, as in our experiments, other selection rules become interesting alternatives to best-first search. In particular, depth-first search has one advantage, when compared to best-first search: it performs less backtracks, which implies faster reoptimization (Atamtürk and Savelsbergh 2005). Therefore, we have tested the two selection rules, best-first and depth-first, for the three approaches, CPLEX, B&C and B&P&C.

The performance analysis depends on the results obtained by the resulting six methods for each problem instance. Table 3 presents the results obtained for the "Easy" instances, which are those that are solved to optimality by the six methods, within the CPU time limit of 3 h. Column "Class" corresponds to the class of problem instances; column "Method" relates to the six methods that result in combining CPLEX, B&C and B&P&C with depth-first and best-first; columns "CPU", "Nodes" and "Variables" display for each method, respectively, the average CPU time in seconds, the total

number of nodes and the number of variables in the models. In this table and the next ones, the number of variables corresponds to the number of variables in the model when the execution stops, since we manipulate a single model throughout the whole execution and do not eliminate any variables during the execution. Note that the slight difference in the number of variables between CPLEX and B&C is due to the addition of the artificial variables in the B&C models, which follows the B&P&C implementation without the column generation step.

These results show that depth-first search performs better than best-first search when used with CPLEX. Indeed, since CPLEX generates cuts only at the root node of the tree, the LP relaxations solved by CPLEX during the B&B enumeration are very similar, irrespective of the selection rule used. The assumption that branching and bounding operations provide the same output, irrespective of the selection rule, is then verified, which is confirmed by the fact that the number of nodes is almost the same for the two selection rules. Because depth-first search performs less backtracks, reoptimization is faster and the CPU times are better for depth-first search. For B&C and B&C&P, the situation is different, since the LP relaxations solved during the enumeration vary significantly depending on the selection rule, because of the continuous addition of cuts and columns during the course of the enumeration. For B&C, the two selection rules give similar results, while for B&P&C, best-first search emerges as a clear winner.

For Class I instances, B&P&C-best is clearly better than any variant of CPLEX and B&C; even though the total number of nodes is larger for B&P&C-best, the small number of variables generated by the method (four times less than the others) more than compensates, translating into an overall algorithm that is about two times faster than the others, on average. For Class II instances, the situation is completely different: the significantly better lower bounds computed by CPLEX lead to an enumerative approach that is an order of magnitude faster than B&C and B&P&C. In addition, the reduction in the number of variables does not payoff this time, since B&P&C is slower than B&C. For Class III-A instances, all methods are equally efficient with average CPU times around 2 seconds. Finally, for Class III-B instances, B&P&C-best is competitive with CPLEX: it is slightly slower than CPLEX-depth, but faster than CPLEX-best. As observed for Class I instances, the large number of nodes incurred by B&P&C is more than compensated by the small number of variables generated by the method. Overall, these results point to the conclusion that B&P&C performs well for problem instances with hundreds of commodities.

To verify this hypothesis, we focus on the "Easy" instances with at least 100 commodities, which are all taken from Classes I and III-B. Table 4 presents the average results obtained on these instances; the columns have the same meanings as in Table 3. On these 41 instances, we see that the total number of nodes is slightly increased when B&P&C-best is performed, compared to CPLEX and B&C methods, but that the CPU time is better, due to the fact that the number of variables is about three times less.

Next, we look at the results obtained with the "Difficult" instances, which are those that are not solved by any of the six methods, within the 3-h CPU time limit. Table 5 presents the results obtained for these instances. Only the best-first search variants of the methods are presented, since depth-first search is not competitive in this case. Indeed, because the enumeration is stopped within the time limit without proving optimality, any selection rule that tends to explore the trees in a breadth-first manner,

Table 4 Results on large-scale "Easy" instances (number of instances in parentheses)

Instances	Method	CPU (s)	Nodes	Variables
$\|K\| \geq 100$ (41)	CPLEX-depth	1,246	3,150	29,983
	CPLEX-best	1,733	3,020	29,983
	B&C-depth	1,580	2,847	30,128
	B&C-best	1,669	2,826	30,128
	B&P&C-depth	1,087	3,722	11,024
	B&P&C-best	1,015	3,317	11,013

Table 5 Results on "Difficult" instances (number of instances in parentheses)

Class	Method	Gap (%)	Nodes	Variables
I (16)	CPLEX-best	0.9	6,792	130,563
	B&C-best	0.9	3,545	130,805
	B&P&C-best	0.7	14,640	30,364
II (2)	CPLEX-best	3.2	16,517	8,401
	B&C-best	7.2	429,172	8,420
	B&P&C-best	7.7	169,185	6,790
III-B (17)	CPLEX-best	1.3	9,900	44,966
	B&C-best	1.4	10,280	45,123
	B&P&C-best	1.3	18,550	22,353

like best-first search does, would deliver final lower bounds that are tighter than those obtained by depth-first search. Thus, in column "Method", we show only the three methods, CPLEX, B&C and B&P&C with best-first search; columns "Gap", "Nodes" and "Variables" display for each method, respectively, the average gap in percentage between the lower and the upper bounds, the total number of nodes and the number of variables in the formulations.

For instances in Classes I and III-B, B&P&C-best provides slightly better lower bounds and final gaps, on average, than the other approaches. Indeed, within the 3-h time limit, the method explores about twice the number of nodes than CPLEX does, which explains why the final gaps are smaller, in spite of the slightly less tight lower bounds computed at the root node. The small number of generated variables explains the efficiency of the B&P&C method: the number of variables is roughly divided by four for Class I instances and by two for Class III-B instances. For Class II instances, CPLEX-best outperforms the other methods because of the quality of its lower bounds, which is reflected in the final gaps. This is in spite of the fact that CPLEX-best explores much less nodes (an order of magnitude less) than the other methods. B&C-best is more efficient than B&P&C-best for these instances, since it is able to explore three times the number of nodes and to reduce the final gaps. Again, these results suggest that B&P&C performs well on instances with hundreds of commodities.

In Table 6, we focus on "Difficult" instances with at least 100 commodities, which are all taken from Classes I and III-B; the columns have the same meanings as in Table 5. On these 31 instances, we see that the final gap is slightly better when B&P&C-best is performed, compared to CPLEX-best and B&C-best, given that the total number of

Table 6 Results on large-scale "Difficult" instances (number of instances in parentheses)

Instances	Method	Gap (%)	Nodes	Variables
$\|K\| \geq 100$ (31)	CPLEX-best	1.1	6,065	91,215
	B&C-best	1.2	4,004	91,424
	B&P&C-best	1.0	12,235	27,390

nodes explored within the 3-hour time limit is significantly increased with B&P&C-best: two times more than CPLEX-best and three times more than B&C-best. The number of variables in the models is roughly divided by three when B&P&C is used.

In addition to the "Easy" and "Difficult" instances, there are nine other instances that are solved by at least one, but not all methods, within the 3-h CPU time limit. Of these nine instances, CPLEX-depth is able to solve to optimality seven of them, B&C-depth two of them and B&P&C-depth three of them. Among these three instances solved by B&P&C-depth, two of them are large-scale Class I instances that cannot be solved by CPLEX-depth. These results confirm the efficiency of the B&P&C algorithm for solving large-scale instances with hundreds of commodities.

Conclusions

In this paper, we have presented a B&P&C algorithm for the multicommodity capacitated fixed-charge network design problem. The RMP solved at each column generation iteration is obtained directly from the compact arc-based model by considering only a subset of the commodity flow variables. The pricing subproblem corresponds to a Lagrangian relaxation of the flow conservation and capacity constraints, leaving in the Lagrangian subproblem only the strong inequalities. A cut generation step based on strong inequalities is also performed. The resulting column-and-row generation procedure is embedded within an enumerative scheme, giving rise to the overall B&P&C algorithm. Our computational experiments show that the B&P&C performs well on large-scale instances with hundreds of commodities. On such instances, the B&P&C algorithm is generally more efficient than a state-of-the-art MIP solver and a B&C algorithm that does not incorporate column generation.

We have explored the relationships between our B&P&C algorithm and other decomposition methods for the problem. Although we have clearly shown the interest of our method by comparing it to a state-of-the-art MIP solver and a B&C algorithm, it would be interesting to perform an extensive analysis of the relative performance of the B&P&C algorithm when compared to other decomposition approaches, in particular Lagrangian-based methods. As mentioned in Sect. 2.6, our algorithm is a special case of the structured Dantzig–Wolfe framework, which has been applied to another multicommodity network design problem Frangioni and Gendron (2013). It would be interesting to extend the approach further to other network design formulations.

Acknowledgments Funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC). This support is gratefully acknowledged. We thank the SCIP team of the Zuse-Institut Berlin (ZIB) for their help in using the SCIP library, especially Tobias Achterberg and Stefan Vigerske. We thank two anonymous referees whose comments have helped us to improve our paper.

References

Aardal K (1998) Capacitated facility location: separation algorithms and computational experience. Math Program 81:149–175

Aardal K, Pochet Y, Wolsey LA (1995) Capacitated facility location: valid inequalities and facets. Math Oper Res 20:562–582

Achterberg T (2009) SCIP: solving constraint integer programs. Math Program Comput 1:1–41

Achterberg T, Koch T, Martin A (2005) Branching rules revisited. Oper Res Lett 33(1):42–54

Atamtürk A (2002) On capacitated network design cut-set polyhedra. Math Program 92:425–437

Atamtürk A, Rajan D (2002) On splittable and unsplittable capacitated network design arc-set polyhedra. Math Program 92:315–333

Atamtürk A, Savelsbergh MWP (2005) Integer-programming software systems. Ann Oper Res 140:67–124

Barahona F (1996) Network design using cut inequalities. SIAM J Optim 6:823–837

Bienstock D, Chopra S, Günlük O, Tsai CY (1998) Minimum cost capacity installation for multicommodity network flows. Math Program 81:177–199

Bienstock D, Günlük O (1996) Capacitated network design-polyhedral structure and computation. INFORMS J Comput 8:243–259

Chouman M, Crainic, TG, Gendron B (2011) Commodity representations and cutset-based inequalities for multicommodity capacitated fixed charge network design. Technical Report CIRRELT-152, CIRRELT, 2011–56

Costa AM, Cordeau J-F, Gendron B, Laporte G (2012) Accelerating Benders decomposition with heuristic master problem solutions. Pesquisa Operacional 32:3–19

Costa AM, Cordeau JF, Gendron B (2009) Benders, metric and cutset inequalities for multicommodity capacitated network design. Comput Optim Appl 42:371–392

Crainic TG, Gendreau M, Farvolden JM (2000) A simplex-based tabu search method for capacitated network design. INFORMS J Comput 12:223–236

Crainic, TG, Frangioni A, Gendron B (1999) Multicommodity capacitated network design. In: Soriano P, Sanso B (eds) Telecommunications network planning. Kluwer Academics Publisher, Dordrecht, pp 1–19

Crainic TG, Frangioni A, Gendron B (2001) Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. Discret Appl Math 112:73–99

Crainic TG, Gendreau M (2002) Cooperative parallel tabu search for capacitated network design. J Heuristics 8:601–627

Crainic TG, Gendron B, Hernu G (2004) A slope scaling/Lagrangean perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design. J Heuristics 10:525–545

Desaulniers G, Desrosiers J, Solomon MM (2005) Column generation. Springer, New York

Frangioni A (2005) About Lagrangian methods in integer optimization. Ann Oper Res 139:163–193

Frangioni A. 2014. http://www.di.unipi.it/~frangio

Frangioni A, Gendron B (2013) A stabilized structured Dantzig–Wolfe decomposition method. Math Program 140:45–76

Frangioni A , Gorgone E (2013) Generalized bundle methods for sum-functions with “easy” components: applications to multicommodity network design. Math Programm (to appear)

Gabrel V, Knippel A, Minoux M (1999) Exact solution of multicommodity network optimization problems with general step cost functions. Oper Res Lett 25:15–23

Günlük O (1999) A branch-and-cut algorithm for capacitated network design problems. Math Program 86:17–39

Gendron B, Crainic TG (1994) Relaxations for multicommodity capacitated network design problems. Technical report, Publication CRT-945, Centre de recherche sur les transports, Université de Montréal,

Ghamlouche I, Crainic TG, Gendreau M (2003) Cycle-based neighbourhoods for fixed charge capacitated multicommodity network design. Oper Res 51:655–667

Ghamlouche I, Crainic TG, Gendreau M (2004) Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. Ann Oper Res 131:109–133

Hewitt M, Nemhauser GL, Savelsbergh MWP (2010) Combining exact and heuristic approches for the capacitated fixed-charge network flow problem. INFORMS J Comput 22:314–325

Holmberg K, Yuan D (2000) A Lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. Oper Res 48:461–481

Ibaraki T (1988) Enumerative approaches to combinatorial optimization. Ann Oper Res 10–11:1–602

Katayama N, Chen M, Kubo M (2009) A capacity scaling heuristic for the multicommodity capacitated network design problem. J Comput Appl Math 232:90–101

Kliewer G, Timajev L (2005) Relax-and-cut for capacitated network design. In: Proceedings of algorithms-ESA 2005: 13th Annual European Symposium on Algorithms. Lecture notes in computer science, vol 3369, pp 47–58

Leung JMY, Magnanti TL (1989) Valid inequalities and facets of the capacitated plant location problems. Math Program 44:271–291

Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. Oper Res 53:1007–1023

Magnanti TL, Mirchandani PB, Vachani R (1993) The convex hull of two core capacitated network design problems. Math Program 60:233–250

Magnanti TL, Mirchandani PB, Vachani R (1995) Modeling and solving the two-facility capacitated network loading problem. Oper Res 43:142–157

Magnanti TL, Wong RT (1984) Network design and transportation planning: models and algorithms. Transp Sci 18:1–55

Muter I, Birbil SI., Bülbül K (2012) Simultaneous column-and-row generation for large-scale linear programs with column-dependent-rows. Math Program 142:47–82

Ortega F, Wolsey LA (2003) A branch-and-cut algorithm for the single commodity uncapacitated fixed charge network flow problem. Networks 41:143–158

Raack C, Koster AMCA, Orlowski S, Wessäly R (2011) On cut-based inequalities for capacitated network design polyhedra. Networks 57:141–156

Rodríguez-Martín I, Salazar-González JJ (2010) A local branching heuristic for the capacitated fixed-charge network design problem. Comput Oper Res 37:575–581

Savelsbergh MWP (1994) Preprocessing and probing techniques for mixed integer programming problems. ORSA J Comput 6:445–454

Sellmann M, Kliewer G, Koberstein A (2002) Lagrangian cardinality cuts and variable fixing for capacitated network design. In Proceedings of Algorithms-ESA 2002: 10th Annual European Symposium on Algorithms. Lecture notes in computer science, vol 2461, pp 845–858