Optimization Letters
DOI 10.1007/s11590-006-0030-5

ORIGINAL PAPER

Optimum synthesis of discrete capacitated networks with multi-terminal commodity flow requirements

Mohamed Haouari · Mehdi Mrad · Hanif D. Sherali

Received: 19 April 2006 / Accepted: 22 September 2006

Abstract Network design problems arise in a wide range of applied areas including telecommunications, computer networks, and transportation. In this paper, we address the following discrete capacitated multi-terminal network design problem. Given a connected digraph $G = (V, A)$, a set of L potential facilities to be installed on each arc, and a set of K multi-terminal (non-simultaneous) commodity flow requirements, the problem is to find a set of facilities to install in order to route the K nonsimultaneous flows while minimizing the total fixed plus variable costs. We describe an exact procedure for solving this problem based on Benders decomposition. Our algorithm includes several features that significantly improve the efficiency of the basic approach. Computational results attest to the efficacy of the proposed algorithm, which can solve medium- to large-scale problems to optimality.

Keywords Network design · Synthesis of capacitated networks · Benders decomposition

M. Haouari (✉) · M. Mrad
Combinatorial Optimization Research Group – ROI, Ecole Polytechnique de Tunisie, BP 743, 2078, La Marsa, Tunisia
e-mail: mohamed.haouari@ept.rnu.tn

M. Mrad
e-mail: mehdi.mrad@ept.rnu.tn

H. D. Sherali
Grado Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA
e-mail: hanifs@vt.edu

1 Introduction

Network design problems arise in a rich variety of applications within the domains of telecommunications, computer networks, and transportation. In this paper, we address the following discrete capacitated multi-terminal network design problem. We are given a (weakly) connected digraph $G = (V, A)$ (i.e., the underlying undirected graph is connected), with $|V| = n$ and $|A| = m$, a set of L of potential facilities to be installed on each arc, and a set of K non-simultaneous multi-terminal commodity flow requirements. Each facility l ($l = 1, \ldots, L$) is capacitated by an upper bound u_l on the amount of flow that it can process. Associated with each commodity k ($k = 1, \ldots, K$) is a requested flow value d_k that must be routed between a specified source s_k and a specified sink t_k. The design cost for each facility l installed on arc (i,j) is $f_{ij}^l \geq 0$ and the variable cost per unit of commodity k is $c_{ij}^k \geq 0$. The problem is to install a set of facilities on each edge as desired such that the K flows can be routed non-simultaneously at a minimum total fixed (design) plus variable (flow) costs. We refer to this problem as the Fixed Charge Network Flow Problem with Multiple Facilities (FCNFMF). An application of the FCNFMF arises in agricultural irrigation systems in arid environments as follows. Given a set of K parcels of land that share a common source of water s_0 for irrigation purposes. Each parcel k ($k = 1, \ldots, K$), belongs to a different grower and requires a water flow of value d_k that must be routed through a pipe network to a specified sink t_k. Given that the region suffers from a constant water deficit, local water authorities allocate irrigation rights to the K growers in turn. The pipes required for designing the irrigation network are available in L different sizes and costs. An important component of the irrigation network design process is the selection of the sizes of the various pipes to be installed so that the flow demand is satisfied while the total installation and operating costs are minimized. Other applications arise in the context of designing multi-terminal communication networks for K users that must at least sustain each user's requirement under dedicated service [7].

The FCNFMF belongs to an important class of network design problems where a trade-off between installation and operating costs is targeted. These problems are particularly attractive because, on one hand, they are notoriously difficult, and on the other hand, they are relevant to a large number of important practical applications in telecommunications, transportation, logistics, location, and production planning. We refer the reader to Magnanti and Wong [13], Minoux [15] and Balakrishnan et al. [2] for general surveys on fixed-charge network design problems. To the best of our knowledge, the FCNFMF has never been addressed in the literature in the foregoing general form. However, many important special cases have been investigated in numerous papers. In particular, if there is just one commodity to be routed between a specified pair of nodes (i.e, $K = 1$), and just one facility type that might be installed (i.e, $L = 1$), then the problem reduces to the well-known fixed charge network flow (FCNF) problem. This problem was investigated in the pioneering papers of Hirsch and Dantzig [8] and Murty [17], who proposed exact optimization approaches. However, most of the subsequent contributions have dealt with approximate solution strategies [5,9–11]. It is noteworthy that if G is a bipartite graph, then the FCNF reduces to the so-called fixed charge transportation problem (FCT) (see [12], and the references therein). A further important, but significantly easier, variant of the FCNFMF is the multi-terminal (non-simultaneous) single-commodity synthesis problem, which has been studied by Gomory and Hu [7]. In this problem, a network needs to be designed that can non-simultaneously sustain some K required single commodity flows. However, no variable flow costs are considered and the design costs vary continuously, where the capacities to be installed are continuous variables. More precisely, for each arc $(i,j) \in A$, if y_{ij} is the capacity assigned to (i,j) then the corresponding design cost is $\gamma_{ij}y_{ij}$, where γ_{ij} is the cost per unit capacity. The authors develop a linear programming approach with a constraint generation scheme to solve this problem. Also, it is noteworthy that in the case where the K flows must be routed *simultaneously,* the problem turns out to be a multicommodity network design problem [16]. Moreover, [14] has investigated the problem of non-simultaneous routing of K muticommodity flows where both installation and operating costs are minimized. We refer to [4] for a recent and exhaustive review of various multicommodity network design problems.

In this paper, we describe an efficient implementation of Benders decomposition [3] for solving the FCNFMF. Since its discovery more than four decades ago, Benders decomposition has been applied to numerous large-scale mixed-integer linear programs. In particular, we cite the seminal application of Geoffrion and Graves [6] on the design of multi-stage distribution networks.

The remainder of this paper is organized as follows. In Sect. 2, we present two valid mathematical programming formulations for the FCNFMF. In Sect. 3, we discuss a tailored Benders decomposition, which is improved in Sect. 4 through the use of several enhancements. In Sect. 5, we report the results of our computational experiments. Finally, some concluding remarks are provided in the last section.

Throughout the paper we shall conform with the following notation. For a node subset $W \subset V$, we define $\delta^+(W) = \{(i,j) \in A : i \in W \text{ and } j \in V \setminus W\}$, and $u(W)$ as the total capacity of the arcs in $\delta^+(W)$. For simplicity, $\delta^+(\{i\})$ is denoted $\delta^+(i)$. Similarly, $\delta^-(W) = \{(i,j) \in A : j \in W \text{ and } i \in V \setminus W\}$. Also, we define $v^+(W) = \{j \in V \setminus W : (i,j) \in \delta^+(W)\}$ and $v^-(W) = \{i \in V \setminus W : (i,j) \in \delta^-(W)\}$.

2 Mathematical programming formulations

A natural mixed-integer programming formulation of the problem can be constructed using continuous flow variables x_{ij}^k for each arc $(i,j) \in A$ and each commodity k ($k = 1,\ldots,K$), and binary variables y_{ij}^l for each arc $(i,j) \in A$ and each facility l ($l = 1,\ldots,L$), representing whether or not facility l is installed on arc (i,j). This yields the following model.

$$\text{FCNFMF:} \quad \text{Minimize} \sum_{k=1}^{K} \sum_{(i,j)\in A} c_{ij}^{k} x_{ij}^{k} + \sum_{l=1}^{L} \sum_{(i,j)\in A} f_{ij}^{l} y_{ij}^{l} \tag{1}$$

subject to:

$$\sum_{j:(i,j)\in A} x_{ij}^{k} - \sum_{j:(j,i)\in A} x_{ji}^{k} = \begin{cases} d_k & \text{if } i = s_k \\ 0 & \text{if } i \in V \setminus \{s_k, t_k\}, \\ -d_k & \text{if } i = t_k \end{cases} \quad \forall i \in V, \forall\, k = 1, \ldots, K, \tag{2}$$

$$\sum_{l=1}^{L} y_{ij}^{l} \leq 1, \quad \forall (i,j) \in A, \tag{3}$$

$$x_{ij}^{k} \leq \sum_{l=1}^{L} u_l y_{ij}^{l}, \quad \forall (i,j) \in A,\ k = 1, \ldots, K, \tag{4}$$

$$x_{ij}^{k} \geq 0, \quad \forall (i,j) \in A,\ k = 1, \ldots, K, \tag{5}$$

$$y_{ij}^{l} \in \{0, 1\}, \quad \forall (i,j) \in A,\ l = 1, \ldots, L. \tag{6}$$

This formulation is self-explanatory and is a straightforward extension of the arc-based formulation of the fixed charge network flow problem. A second formulation, which will be referred to as the cut-based formulation, is derived from the well-known *max-flow-min-cut theorem* [3]. Indeed, an immediate consequence of this theorem is that a necessary and sufficient condition for a feasible flow of value d_k between s_k and t_k is that for each cut $\delta^{+}(W)$, with a corresponding installed capacity $u(W)$, and induced by a subset $W \subset V$ satisfying $s_k \in W$ and $t_k \in \bar{V} \equiv V \setminus W$, we have

$$u(W) \geq d_k. \tag{7}$$

Now, assume that there are λ_k different cuts in G separating s_k and t_k and denote by $W_p^k (p = 1, \ldots, \lambda_k)$ the corresponding subsets. Therefore, it follows from (7) that the set of installed facilities is feasible for commodity k if and only if

$$\sum_{(i,j)\in \delta^{+}(W_p^k)} \sum_{l=1}^{L} u_l y_{ij}^{l} \geq d_k, \quad \forall p = 1, \ldots, \lambda_k. \tag{8}$$

Hence, an alternative valid formulation of the problem is given as follows, which is stated in a naturally decomposed form (in lieu of a consolidated optimization model) for the sake of convenience in developing a Benders approach in the sequel.

$$\text{Minimize} \sum_{l=1}^{L} \sum_{(i,j)\in A} f_{ij}^{l} y_{ij}^{l} + \sum_{k=1}^{K} g_k(y) \tag{9}$$

subject to

$$\sum_{(i,j)\in\delta^+(W_p^k)} \sum_{l=1}^{L} u_l y_{ij}^l \geq d_k, \quad \forall p = 1,\ldots,\lambda_k, \; k = 1,\ldots,K, \tag{10}$$

$$\sum_{l=1}^{L} y_{ij}^l \leq 1, \quad \forall (i,j) \in A, \tag{11}$$

$$y_{ij}^l \in \{0,1\}, \quad \forall (i,j) \in A, \quad l = 1,\ldots,L, \tag{12}$$

where for each $k = 1,\ldots,K$, we have

$$g_k(y) \equiv \text{Minimum} \sum_{(i,j)\in A} c_{ij}^k x_{ij}^k \tag{13}$$

subject to:

$$\sum_{j:(i,j)\in A} x_{ij}^k - \sum_{j:(j,i)\in A} x_{ji}^k = \begin{cases} d_k & \text{if } i = s_k, \\ 0 & \text{if } i \in V \setminus \{s_k, t_k\}, \;\; \forall i \in V, \\ -d_k & \text{if } i = t_k, \end{cases} \tag{14}$$

$$x_{ij}^k \leq \sum_{l=1}^{L} u_l y_{ij}^l, \quad \forall (i,j) \in A, \tag{15}$$

$$x_{ij}^k \geq 0, \quad \forall (i,j) \in A. \tag{16}$$

3 Benders decomposition

A natural way to tackle the problem defined by (9)–(16) is to use Benders decomposition. To that aim, using duality, we rewrite $g_k(y)$ as

$$g_k(y) = \text{Maximum } d_k(\alpha_{s_k}^k - \alpha_{t_k}^k) - \sum_{(i,j)\in A} \left(\sum_{l=1}^{L} u_l y_{ij}^l \right) \beta_{ij}^k \tag{17}$$

subject to:

$$\alpha_i^k - \alpha_j^k - \beta_{ij}^k \leq c_{ij}^k, \quad \forall\, (i,j) \in A, \tag{18}$$

$$\beta_{ij}^k \geq 0, \quad \forall\, (i,j) \in A. \tag{19}$$

Now, let $(\alpha_{ih}^k, \beta_{ijh}^k)$ for $i \in V$, $(i,j) \in A$, and $h = 1,\ldots,H_k$, be the extreme points of the polyhedron defined by (18)–(19) and denote $a_h^k = d_k(\alpha_{s_k h}^k - \alpha_{t_k h}^k)$ for $h = 1,\ldots,H_k$, and $b_{ijlh}^k = u_l \beta_{ijh}^k$ for $(i,j) \in A, h = 1,\ldots,H_k$. Then, given a y

that is feasible to (10)–(12) (or its continuous relaxation), $g_k(y)$ can be stated as

$$g_k(y) = \underset{h=1,\ldots,H_k}{\text{Maximum}} \;\; a_h^k - \sum_{(i,j)\in A} \sum_{l=1}^{L} b_{ijlh}^k y_{ij}^l. \tag{20}$$

This yields the Benders reformulation of the problem defined by (9)–(16) as follows.

$$\text{Minimize} \sum_{(i,j)\in A} \sum_{l=1}^{L} f_{ij}^l y_{ij}^l + \sum_{k=1}^{K} \eta_k \tag{21}$$

subject to:

$$\sum_{(i,j)\in\delta^+(W_p^k)} \sum_{l=1}^{L} u_l y_{ij}^l \geq d_k, \quad \forall p = 1,\ldots,\lambda_k, \quad k = 1,\ldots,K, \tag{22}$$

$$\sum_{l=1}^{L} y_{ij}^l \leq 1, \quad \forall (i,j) \in A, \tag{23}$$

$$\eta_k \geq a_h^k - \sum_{(i,j)\in A} \sum_{l=1}^{L} b_{ijlh}^k y_{ij}^l, \quad \forall h = 1,\ldots,H_k,\ k = 1,\ldots,K, \tag{24}$$

$$y_{ij}^l \in \{0,1\}, \quad \forall (i,j) \in A,\ l = 1,\ldots,L, \tag{25}$$

$$\eta_k \geq 0, \quad \forall k = 1,\ldots,K, \tag{26}$$

where the implied nonnegativity constraints (26) are added for the sake of algorithmic convenience. We now adopt a constraint generation procedure where a relaxation of the problem, having a restricted number of constraints (22) and (24), is solved at each iteration, as outlined below.

3.1 Benders decomposition procedure (basic version)

Step 1: *Initialization* Let P_1 be the problem defined by (21), (23), (25), and (26). Set $q = 0$.
Step 2: *MIP-solver* Set $q \leftarrow q + 1$. Solve P_q using an MIP solver. Let $(\bar{y}, \bar{\eta})$ be an optimal solution to P_q.
Step 3: *Feasibility cut identification* For each commodity k ($k = 1,\ldots,K$), find a feasibility constraint $\sum_{(i,j)\in\delta^+(W_p^k)} \sum_{l=1}^{L} u_l y_{ij}^l \geq d_k, \quad p \in \{1,\ldots,\lambda_k\}$ which is violated by $\bar{y}$ or verify that no such inequality exists.
Step 4: *Feasibility test* If one or more violated feasibility constraints are found in Step 3, then define P_{q+1} to be P_q amended by the violated feasibility constraints and go to Step 2. Else, go to Step 5.

Step 5: *Optimality cut identification* For each commodity k ($k = 1, \ldots, K$), find an optimality constraint $\eta_k \geq a_h^k - \sum_{(i,j)\in A} \sum_{l=1}^{L} b_{ijlh}^k y_{ij}^l$, which is violated by $(\bar{y}, \bar{\eta}_k)$ (i.e, find (a_h^k, b_{ijlh}^k) such that $a_h^k - \sum_{(i,j)\in A} \sum_{l=1}^{L} b_{ijlh}^k \bar{y}_{ij}^l > \bar{\eta}_k$), or verify that no such inequality exists.
Step 6: *Optimality test* If one or more violated optimality constraints are found in Step 5, then define P_{q+1} to be P_q amended by the violated optimality constraints and go to Step 2. Else, stop.

Step 3 requires the solution of K independent subproblems of the form:
(**SP1**$_k$) – *Given $\bar{y}$, find $W \subset V$ satisfying $s_k \in W$ and $t_k \in V \setminus W$, such that*

$$\sum_{(i,j)\in\delta^+(W)} \sum_{l=1}^{L} u_l \bar{y}_{ij}^l < d_k,$$

or verify that no such subset exists.
Hence, $SP1_k$ amounts to finding

$$W_k^* = \underset{\substack{W\subset V:\\ s_k\in W \text{ and } t_k\in\bar{W}}}{\operatorname{argmin}} \left\{ \sum_{(i,j)\in\delta^+(W)} \sum_{l=1}^{L} u_l \bar{y}_{ij}^l \right\}.$$

This problem amounts to finding a minimum cut separating s_k and t_k on the network G, where the capacity of each arc $(i,j) \in A$ is $\sum_{l=1}^{L} u_l \bar{y}_{ij}^l$. Let φ_k denote the value of the maximum flow between s_k and t_k using these arc capacities. Due to the max-flow-min-cut theorem [1], if $\varphi_k < d_k$ then the node subset corresponding to the minimal cut solves $SP1_k$ (Gomory and Hu, 1962).

Moreover, Step 5 requires the solution of K independent subproblems of the form:
(**SP2**$_k$) – *Given $(\bar{y}, \bar{\eta})$, find $h \in \{1, \ldots, H_k\}$ such that*

$$a_h^k - \sum_{(i,j)\in A} \sum_{l=1}^{L} b_{ijlh}^k \bar{y}_{ij}^l > \bar{\eta}_k,$$

or verify that no such inequality exists.
Hence, $(SP2_k)$ amounts to solving

$$g_k(y) = \underset{h=1,\ldots,H_k}{\text{Maximum}} \left\{ a_h^k - \sum_{(i,j)\in A} \sum_{l=1}^{L} b_{ijlh}^k y_{ij}^l \right\},$$

which involves solving a minimum cost flow problem where d_k units of flow must be sent from s_k to t_k on the network G, and where the capacity of each arc $(i,j) \in A$ is $\sum_{l=1}^{L} u_l \bar{y}_{ij}^l$. It is noteworthy that, since $\bar{y}$ satisfies (22) and (23), then $SP2_k$ is necessarily feasible.

The maximum flow problem and the minimum cost flow problem are solvable by efficient polynomial-time algorithms [1]. For instance, the so-called highest-label preflow-push algorithm solves the maximum flow problem in $O(n^2\sqrt{m})$ time. Also, the enhanced capacity scaling algorithm solves the minimum cost flow problem in $O((m\log n)(m+n\log n))$ time.

At this point, it is worth mentioning that the above described solution procedure remains a valid approach for solving a slightly more general version of the FCNFMF where multiple facilities of the same type might be installed on the same arc. Indeed, only minor changes are required for tackling this variant. More precisely, Constraints (3), (11), and (23) should be dropped, and Constraints (6), (12), and (25) should be replaced by

$$y_{ij}^{l} \in \mathbb{N}, \quad \forall (i,j) \in A, \ \ l = 1, \ldots, L. \tag{6'}$$

4 Enhancements

In order to improve the efficiency of the foregoing approach, we propose several further enhancements that are described below.

4.1 Solving the linear relaxation and integration of a heuristic

The main computational effort of the proposed approach is performed in Step 2, which might be too time consuming since it requires the exact solution of an $\mathcal{NP}$-hard problem. Therefore, in order to alleviate the computational burden we modify Step 2 in the following way. Given a Problem P_q, we first start by solving its linear relaxation, which is denoted by LP_q. Then, we perform a feasibility test in order to check whether the continuous optimal solution $\tilde{y}$ violates one or more feasibility cuts. If violated cuts are found, then they are appended to P_q, and its linear relaxation is re-solved. Otherwise, if no violated cuts are found and $\tilde{y}$ is not integer, then P_q is solved by means of a fast heuristic. (In our implementation, an approximate binary solution is found by using a truncated exact optimization algorithm. The algorithm is stopped as soon as it delivers a feasible solution within a percentage relative gap of 1%.) Let $\hat{y}$ denote the resulting approximate feasible binary solution to P_q. If $\hat{y}$ violates one or more cuts, then these cuts are sequentially appended to P_q, which is then again solved approximately. Otherwise, P_q is solved exactly. If the resulting solution y^* violates one or more cuts, then these cuts are appended to P_q, and this augmented problem is again processed as above by first solving it heuristically. Otherwise, y^* is output as an optimal solution.

4.2 Generation of initial feasibility cuts

Instead of starting from scratch, in a preprocessing step, some initial feasibility cuts are generated. For each commodity k, we generate the cuts corresponding to the subsets $\{W^k_\rho, \rho = 1, \ldots, \mu_k\}$ using the following simple recursive procedure:

```
Phase 1 Forward recursion
1.1 Set $W^k_1 = \{s_k\}$, $\rho = 1$,
1.2 While $(v^+(W^k_\rho) \neq \{t_k\})$
Begin
    1.2.1 $W^k_{\rho+1} = (v^+(W^k_\rho)\backslash\{t_k\}) \cup (W^k_\rho)$
    1.2.2 $\rho \leftarrow \rho + 1$
End (While)
Phase 2 Backward recursion
2.1 Set $\bar{W}^k_\rho = \{t_k\}$
2.2 While $(v^-(\bar{W}^k_\rho) \neq \{s_k\})$
Begin
    2.2.1 $\bar{W}^k_{\rho+1} = (v^-(\bar{W}^k_\rho)\backslash\{s_k\}) \cup (\bar{W}^k_\rho)$
    2.2.2 $\rho \leftarrow \rho + 1$
End (While)
```

4.3 Appending path-connectivity constraints

Let $G(y) = (V, A(y))$ denote the graph that is obtained from G by removing all the arcs $(i,j) \in A$ such that $\sum_{l=1}^{L} y^l_{ij} = 0$. Clearly, if y is an optimal solution then the only leaves that $G(y)$ might contain (discarding isolated nodes) are sources and/or sinks. Hence, a valid pair of inequalities is

$$\sum_{l=1}^{L} y^l_{ij} - \sum_{w\in\delta^-(i)} \sum_{l=1}^{L} y^l_{wi} \leq 0, \quad \forall (i,j) \in A, \;\; i \in V \setminus \{s_1, s_2, \ldots, s_K\}, \tag{27}$$

$$\sum_{l=1}^{L} y^l_{ij} - \sum_{w\in\delta^+(j)} \sum_{l=1}^{L} y^l_{jw} \leq 0, \quad \forall (i,j) \in A, \;\; j \in V \setminus \{t_1, t_2, \ldots, t_K\}. \tag{28}$$

Constraints (27) and (28) guarantee that if $\sum_{l=1}^{L} y^l_{ij} = 1$ then for the designated respective indices, we must have $\sum_{w\in\delta^-(i)} \sum_{l=1}^{L} y^l_{wi} \geq 1$ and $\sum_{w\in\delta^+(j)} \sum_{l=1}^{L} y^l_{jw} > 1$. It is noteworthy that since the initial feasibility cuts are included in the model, then every feasible solution also satisfies

$$\sum_{j\in\delta^+(s_k)} \sum_{l=1}^{L} y^l_{s_k,j} \geq 1 \quad \text{and} \quad \sum_{i\in\delta^-(t_k)} \sum_{l=1}^{L} y^l_{i,t_k} \geq 1 \quad \text{for } k = 1, \ldots, K. \tag{29}$$

Hence, constraints (27) and (28) guarantee the existence of a path between every pair (s_k, t_k). In the sequel, we refer to these constraints as path-connectivity constraints. In our implementation, path-connectivity constraints are not included *a priori* in the original model, but only violated constraints are iteratively generated "on the fly" and appended to the model.

5 Computational experiments

In order to assess the practical performance of the proposed approach, we have coded it in Microsoft Visual C++ (version 6.0) in concert with the CPLEX 9.0 solver. All the computational experiments were carried out on a Pentium IV 3.2 GHz Personal Computer with 3.0 GB RAM.

The test-bed we have used consists of randomly generated instances. Each instance, characterized by the number of nodes n, the number of arcs m, the number of commodities K, and the number of facilities L, is constructed in the following way. The coordinates of the nodes are integers chosen randomly from the interval [1, 10]. The corresponding graph is initialized with a random spanning tree (which ensures its connectedness), then additional edges are added randomly. Finally, each edge is replaced by a pair of oppositely directed arcs. Let θ_{ij} denote the Euclidean distance between two points i and j. If the arc $(i,j) \in A$, then the corresponding variable cost is computed as $c_{ij}^k = \lceil \sqrt{\theta_{ij}} \rceil$ (where $\lceil a \rceil$ represents the smallest integer that is greater than or equal to a.) (We also attempted generating problems having variable costs that depend on k according to $c_{ij}^k = \lceil \sqrt{k + \theta_{ij}} \rceil$, $\forall (i,j) \in A$, $k = 1, \ldots, K$, for $K = 5$. All these problems were solved to optimality similar to the experience reported below; hence, for the sake of brevity, we suppress these results here.) Also, the corresponding fixed charge associated with facility l having capacity u_l is computed as $f_{ij}^l = \lceil \theta_{ij} \sqrt{10 u_l} \rceil$. Note that the fixed charge of a facility is a concave function of its capacity, which is compatible with the economy of scale phenomenon. For all the generated instances, we set $L = 3$, and for each arc of the graph, the installed capacity is selected in $U = \{10, 20, 40\}$. For each commodity k ($k = 1, \ldots, K$), the source s_k and the sink t_k are selected randomly, and the demand d_k, $\forall$, is set equal to $d_k = \lceil 0.3\Phi_{\max} \rceil$, where $\Phi_{\max}$ is the value of the maximal flow between s_k and t_k with respect to the intermediate capacity (i.e., $u_l = 20$).

The results are summarized in Table 1. For each instance, we report n : number of nodes, m : number of arcs, *Sol* : value of the optimal solution, *Time* : total CPU time in seconds, FC : number of generated feasibility cuts, OC : number of generated optimality cuts, and PCC : number of generated path-connectivity cuts.

From Table 1, we observe that the proposed approach performs remarkably well, being able to solve exactly relatively large-size instances within a reasonable CPU effort. Indeed, all of the instances, but one, were solved to optimality.

We also tested this methodology on an important special case, namely the fixed charge network flow problem (FCNF). The results are displayed in Table 2. Interestingly, we observe from this table, that large instances of the FCNF having up to 500 nodes and 2,000 arcs were solved to optimality (except that one

Table 1 Computational performance on FCNFMF instances

Instance	n	m	K	Time	Sol	FC	OC	PCC
A01	10	30	5	2.403	842	60	12	42
A02	10	30	10	12.798	1,286	126	25	23
A03	10	40	5	1.211	742	54	9	23
A04	10	40	10	23.043	1,313	110	23	13
A05	10	50	5	0.18	566	42	3	7
A06	10	50	10	12.528	1,224	113	18	16
A07	10	60	5	0.14	650	42	2	5
A08	10	60	10	18.877	1,384	122	24	18
A09	20	50	5	5.317	1,042	135	11	88
A10	20	50	10	10.985	1,502	189	16	69
A11	20	80	5	24.665	780	118	18	82
A12	20	100	5	99.713	784	115	14	106
A13	50	150	5	9.834	1,067	148	22	240
A14*	50	200	5	–	–	–	–	–
A15	50	250	3	29.161	746	76	25	234
A16	100	250	3	30.644	946	89	66	316
A17	100	300	3	4.917	855	73	8	243
A18	100	400	3	99.062	1,136	89	27	381
A19	100	500	3	2.794	766	58	8	109

(*) This instance remained unsolved after reaching the 3,600 s CPU time limit

instance remained unsolved after 3,600 s). To the best of our knowledge, the solution of such large instances has never been previously reported in the literature.

In order to investigate the impact of the proposed enhancements, we ran the basic version of the algorithm that is described in Section 3 on the FCNF instances. The results are summarized in Table 3. In this table, each entry of the columns Time_ratio, FC_ratio, and OC_ ratio, respectively represents the ratio of the CPU time, the number of feasibility cuts, and the number of optimality cuts, obtained with the basic version with respect to that for the enhanced one. Examining this table, we see that for all instances, except for the four ones that required the shortest computing times (less than 0.15 s), the CPU time as well as the number of generated cuts for the basic version increased dramatically. Also, five instances remained unsolved after reaching the maximum CPU time limit (3,600 s) using the basic version (instead of just one for the enhanced one).

Finally, in order to get a better insight into the efficacy of the proposed approach, we compared it with a state-of-the-art MIP solver. To that aim, we used CPLEX 9.0 to solve the arc-based formulation of FCNFMF for instances having installation costs only. For each instance, Table 4 reports the ratio (Time_ratio) of the CPU time required by the MIP solver to that for the proposed algorithm (note that we have again set the maximum CPU time limit to 3,600 s.) From this table, we see that all the 18 instances have been solved by the proposed algorithm within a moderate CPU time. In contrast, only the five smallest instances were solved to optimality by CPLEX, and that too, requiring significantly greater CPU effort.

Table 2 Computational performance on FCNF instances

Instance	n	m	Time	Sol	FC	OC	PCC
B01	50	150	0.109	97	18	2	1
B02	50	200	0.063	64	11	2	1
B03	50	300	1.000	476	75	70	131
B04	50	400	0.078	126	10	3	7
B05	100	300	4.406	529	110	97	204
B06	100	400	0.594	552	47	36	145
B07	100	600	0.141	299	14	5	15
B08	200	600	4.218	456	80	67	202
B09	200	800	183.906	502	643	706	876
B10	200	1,000	0.328	286	18	7	33
B11	200	1,200	0.485	345	21	11	64
B12	300	800	1.188	669	42	24	120
B13	300	1,000	202.938	706	524	511	627
B14	300	1,200	2.875	561	116	103	262
B15*	500	1200	–	–	275	252	836
B16	500	1,600	2.594	457	76	57	230
B17	500	2,000	399.875	553	178	164	810

(*) This instance remained unsolved after reaching the 3,600 s CPU time limit

Table 3 Comparison of the performance of the basic version with respect to the enhanced version

Instance	n	m	Solved	Time_ratio	FC_ratio	OC_ratio
B01	50	150	Yes	0.716	1.000	2.000
B02	50	200	Yes	0.746	1.000	2.000
B03	50	300	Yes	26.765	2.680	2.871
B04	50	400	Yes	0.795	1.200	2.333
B05	100	300	Yes	86.507	4.591	0.021
B06	100	400	Yes	125.184	5.787	7.417
B07	100	600	Yes	0.993	1.429	2.600
B08	200	600	No	>853.485	>13.221	–
B09	200	800	No	>19.261	>1.485	–
B10	200	1,000	Yes	3.003	3.278	7.143
B11	200	1,200	Yes	3.705	2.714	4.455
B12	300	800	Yes	12.942	5.619	9.542
B13	300	1,000	No	>17.821	>1.748	–
B14	300	1,200	Yes	150.609	4.31	0.029
B15	500	1,200	No	–	–	–
B16	500	1,600	Yes	64.479	5.026	0.035
B17	500	2,000	No	>9.002	>5.685	–

6 Conclusion

We have presented an exact algorithm based on Benders decomposition for solving the Fixed Charge Network Flow Problem with Multiple Facilities. The proposed approach includes several distinctive algorithmic features. In particular, it is based on efficiently coordinating the solution of the LP relaxation, heuristics, and MIP solution, while incorporating Benders cuts along with a class of path-connectivity cuts. Computational results attest to the efficacy

Table 4 Comparison of the performance of the proposed algorithm with respect to a general-purpose MIP solver

Instance	n	m	K	Sol	Time	CPLEX_Time	Time_ratio
C01	10	30	45	707	1.11	3.953	3.561
C02	10	40	45	738	1.453	16.343	11.247
C03	10	50	45	680	1.547	10.875	7.029
C04	10	60	45	770	1.438	15.406	10.713
C05	15	50	105	1,220	4.328	–	–
C06*	20	80	190	1,674	5.297	–	–
C07*	30	120	100	2,267	49.016	–	–
C08*	30	120	200	2,380	98.219	–	–
C09*	30	120	300	2,400	208.547	–	–
C10*	40	140	100	3,016	771.406	–	–
C11*	40	140	200	3,199	945.125	–	–
C12*	40	140	250	3,229	607.047	–	–
C13*	50	160	100	3,932	257.281	–	–
C14*	50	160	200	4,157	863.422	–	–
C15*	50	160	250	4,157	649.906	–	–
C16*	100	300	5	875	1694.047	–	–
C17*	100	400	5	***	> 3600	–	–
C18*	100	500	5	888	347.828	–	–

(*) This instance remained unsolved by CPLEX 9.0 after reaching the 3600 seconds CPU time limit

of the proposed algorithm, which can solve to optimality FCNFMF instances having up to 100 nodes and 500 edges. In addition, large-scale instances of the well-studied FCNF having up to 500 nodes and 2,000 edges were solved to optimality. Furthermore, we have provided empirical evidence that the proposed approach consistently outperforms both the basic Benders decomposition algorithm as well as a state-of-the-art MIP solver. Indeed, this lattermost alternative solution strategy failed to solve 13 out of the 18 instances that were successively solved by the proposed algorithm. On the other hand, the basic Benders algorithm failed to solve 5 out of 17 instances that were solved to optimality by the proposed approach, where the latter was able to deliver solutions with more than two orders of magnitude faster than the basic version in some cases.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms and Applications. Prentice Hall, Upper River (1993)
2. Balakrishnan, A., Magnanti, T.L., Mirchandani, P.: Network design. In: Dell'Amico, M., Maffioli, F., Martello S. (eds.) Annotated Bibilographies in Combinatorial Optimization. pp. 311–334. Wiley, New York (1997)
3. Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. Numer. Math. **4**, 238–252 (1962)
4. Costa, A.M.: A survey on benders decomposition applied to fixed-charge network design problems. Comput. Oper. Res. **32**, 1492–1450 (2005)
5. Eksioglu, B., Eksioglu, S.D., Pardalos, P.M.: Solving large scale fixed charge network flow problem. In: Mangeri, A., Giannesi, F., Pardalos P.M., (Eds.) Equilibrium Problems and Variational Models. Kluwer, Dordrecht (2001)

6. Geoffrion, A.M., Graves, G.W.: Multicommodity distribution system design by Benders decomposition. Manage. Sci. **20**, 822–844 (1974)
7. Gomory, R.E., Hu, T.C.: An application of generalized linear programming to network flows. SIAM J. Appl. Math. **10**, 260–283 (1962)
8. Hirsh, W.M., Dantzig, G.B.: The fixed charge problem. Naval Res. Logistics Q. **15**, 413–424 (1968)
9. Khang, D.B., Fujiwara, O.: Approximate solutions of capacitated fixed-charge minimum cost network flow problems. Networks **21**, 689–704 (1991)
10. Kim, D., Pardalos, P.M.: A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. Oper. Res. Lett. **24**, 195–203 (1999)
11. Kim, H.J., Hooker, J.N.: Solving fixed-charge network flow problems with a hybrid optimization and constraint programming approach. Ann. Oper. Res. **115**, 95–124 (2002)
12. Lamar, B.W., Wallace, C.A.: Revised-modified penalties for fixed charge transportation problems. Manage. Sci. **43**, 1431–1436 (1997)
13. Magnanti, T.L., Wong, R.T.: Network design and transportation planning: models and algorithms. Transportation Sci. **18**, 1–55 (1984)
14. Minoux, M.: Optimum synthesis of a network with non-simultaneous multicommodity flow requirements. Ann. Discrete Math. **11**, 269–277 (1981)
15. Minoux, M.: Network synthesis and optimum network design problems: models, solution methods and applications. Networks **19**, 313–360 (1989)
16. Minoux, M.: Discrete cost multicommodity network optimization problems and exact solution methods. Ann. Oper. Res. **106**, 19–46 (2001)
17. Murty, K.G.: Solving the fixed charge problem by ranking the extreme points. Oper. Res. **16**, 268–279 (1968)