Proceedings of the XI Brazilian Congress on Automatic Control (XI CBA)
São Paulo, pp. 359–364, September, 1996

A BRANCHING STRATEGY FOR UNCAPACITATED FIXED-CHARGE NETWORK FLOW PROBLEMS

F. R. B. Cruz[a,†,‡], G. R. Mateus[b,§], J. MacGregor Smith[a]

[a]*Department of Mechanical and Industrial Engineering,*
The University of Massachusetts,
Amherst, MA 01003, USA
E-mail: {fcruz,jmsmith}@ecs.umass.edu

[b]*Departamento de Ciência da Computação,*
Universidade Federal de Minas Gerais,
31270-010 - Belo Horizonte - MG, Brazil
E-mail: mateus@dcc.ufmg.br

Abstract: Given a directed network, the uncapacitated fixed-charge network flow problem is that of finding a minimum-cost arc combination that provides flows from the supply nodes to the demand nodes. Associated with all arcs there are two costs involved, the fixed charge of using the arc and the variable cost depending on the amount of flow the arc actually carries. This generic model has applications for problems of distribution, transportation, communication, and routing. The purpose of this paper is to present an improved branching strategy to solve the problem using branch-and-bound algorithms.

Key words: Network Optimization, Fixed-charge Problems, Steiner Problems in Graphs.

1. INTRODUCTION

The uncapacitated fixed-charge network flow (UFNF) problem represents an important class of mixed-integer programming problems. The problems are defined on a digraph $\mathcal{D} = (N, A)$, where N is the set of nodes and A is the set of arcs. One of the costs involved is the fixed cost of using an arc to send flow and the other is a variable cost dependent on the amount of flow sent through the arc. The objective is to determine a minimum cost arc combination that provides flows from certain supply nodes to a collection of demand nodes, possibly using intermediate *Steiner* or transshipment nodes. A single-supply-node instance of the problem is depicted in Figure 1.

This is an $\mathcal{NP}$-hard optimization problem [6] generalizing among others the Steiner problem in graphs [13]. This generic model has applications for problems of distribution, transportation and communication. It is also useful for certain routing

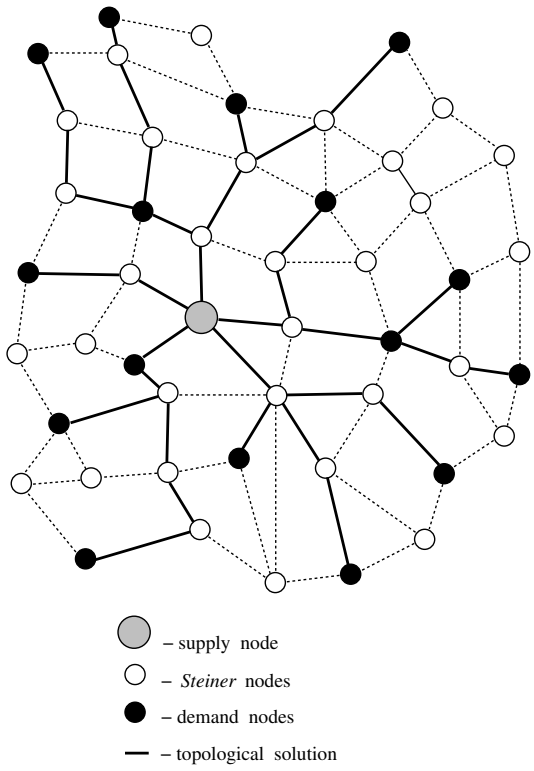

Figure 1: A Fixed-charge Network Flow Problem

†To whom all correspondences should be addressed. Address: Caixa Postal 702, Departamento de Ciência da Computação - UFMG, 30161-970 - Belo Horizonte - MG, Brazil.

‡Visiting research scholar supported by the *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq), Brazil, under grant CNPq 201046/94-6.

§Supported by CNPq and FAPEMIG, Brazil.

problems where the network is already in existence. Besides being an important model by itself, several special cases of the UFNF problem are of substantial interest. A simple way to obtain special cases is to restrict the network structure, *e.g.* as in the transportation problem.

One example of an application closely related to the UFNF problem is in the field of distribution systems, *e.g.* the problem of designing offshore natural-gas pipeline systems [22]. The central separation plant is located on land and typically serves multiple offshore under-water gas wells. The problem is how to transport gas from the offshore wells to the separation plant at minimum cost. An important characteristic of gas pipelines systems is that the construction cost consists of fixed components that are independent of the amount of flow and variable components which are proportional to the amount of flow.

Another application in communication network design is treated in detail by [17]. The switching center network problem described there consists of looking for a topology on the urban street network that minimizes the total cost of cables and subterranean piping infrastructure necessary to link a telephone center and its subscribers. Again, the fixed cost — represented by the subterranean infrastructure — and the flow-dependent cost — represented by the cables — occur simultaneously for each arc. Actually, the network depicted in Figure 1 was taken from this application, which is an illustration of a local switching center network problem. The dotted arcs represent important streets and the bold ones represent the topological solution. The supply node represents a local switching center, the demand nodes represent groups of subscribers, and the *Steiner* nodes represent the street intersections. More details on this problem may be found in [17].

Finally, even considering those cases where the network topology is known in advance, important routing problems emerge. Suppose, for example, that one specific node in the network has to send messages at minimum cost addressed to a specific collection of other nodes in the network. The UFNF model is applicable since it is a reasonable assumption that an initial set up cost is associated with each link selected independent of the flow as well as a variable cost dependent on how much information has to be sent.

Some experimental work concerning exact and approximate solutions for the UFNF problems and their special cases have been done previously. In [22], an analysis of offshore natural-gas systems was done with the cost model being simplified including only fixed costs. In [17], the model studied was even more complex than the UFNF problem, presenting some additional features. However, only heuristic procedures and local optimization techniques were considered. The special case without *Steiner* nodes was treated by [16] and by [19]. In the former, an exact branch-and-bound algorithm combined with Benders cuts was studied, and in the latter, a set of heuristic procedures based on Lagrangean relaxation techniques was developed. In [2, 5], some special cases were solved using a branch-and-bound algorithm with fractional cutting-planes. Previously, we have studied *ADD* and *DROP* heuristic approaches [7, 20] as well as simplified branch-and-bound algorithms [8, 9] to solve the general UFNF problems and these algorithms have performed well in practice.

The branch-and-bound approach is known to be inefficient because of its combinatorial explosive behavior and it is often only acceptable for small problem instances. The use of improved branching techniques is justified because they may speed-up the algorithm consequently enlarging the size of manageable instances. Keeping in mind the importance of the UFNF problems and the necessity of solving large instances generated by real word problems, we are proposing procedures designed to improve the effectiveness of branch-and-bound algorithms in conventional computer systems. However, we are aware that there always will be cases where an exact approach may be impossible no matter what systems or algorithms are utilized.

The outline of this paper is as follows. In Section 2, the mathematical programming formulation of the UFNF problem will be presented. In Section 3, we shall discuss the solution methods we are proposing. All proposed algorithms were implemented and our experimental results using this implementation are reported in Section 4. Section 5 closes this work with final remarks, open questions, and the presentation of some possible extensions.

2. PROBLEM FORMULATION

A natural generalization of the UFNF problem is the capacitated fixed-charge network flow (CFNF) problem. Although the CFNF problem is very difficult to be solved, it can be represented by a surprisingly compact mathematical programming formulation [21]:

(M)**:**

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

s.t.:

$$\sum_{j\in\delta^+(i)} x_{ij} - \sum_{j\in\delta^-(i)} x_{ji} = b_i, \quad \forall \quad i \in N, \tag{2}$$

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

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

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

where $\mathcal{D} = (N, A)$ is a digraph, N is the set of nodes, A is the set of arcs, $\delta^+(i) = \{j|(i,j) \in A\}$, $\delta^-(i) = \{j|(j,i) \in A\}$, $b_i > 0$ (< 0) is the supply (demand) at node i, f_{ij} is the fixed cost of having flow on arc (i,j), c_{ij} is the variable cost per unit of flow on arc (i,j), and u_{ij} is the capacity of arc (i,j). It is noticeable that the only difference between the CFNF problem and the linear minimum-cost network flow (MCNF) problem is that, in the former, if the flow is positive, *i.e.* $x_{ij} > 0$, then its cost is $c_{ij}x_{ij} + f_{ij}$. The capacity constraints (3) ensure that characteristic. That simple difference transforms the polynomially solvable MCNF problem into the $\mathcal{NP}$-hard CFNF problem.

As pointed out by [21], a necessary condition for feasibility, assumed throughout this work, is that $\sum_{i \in N} b_i = 0$. Additionally, it is assumed that all problems are single-supply-node. The fixed cost f_{ij} must be non-negative for all arcs (i,j), since otherwise one could set y_{ij} to 1 and eliminate it from the problem. On the other hand, the variable cost c_{ij} is unrestricted. However, to ensure that the objective function is bounded from below, it is assumed that there are no negative-cost directed cycles with respect to c_{ij}.

An important simplification we are considering here concerns the capacity constraints. If the u_{ij} is sufficiently large, say $u_{ij} \geq \frac{1}{2}\sum_{i \in N} |b_i|$, the capacity constraints only force the inclusion of the fixed cost in the objective function when the flow is positive. The problems under such assumption are called *uncapacitated* and these are the only problems treated in this work.

3. SOLUTION METHOD

The branch-and-bound algorithm is a well-known procedure to solve $\mathcal{NP}$-hard problems, computationally inefficient because of its exponential worst-case time complexity, $O(2^{|A|})$, but acceptable in practice for small sized problem instances. Basically, three statements are present in branch-and-bound algorithms: computation of lower and upper bounds and a strategy of choosing the branching variables.

3.1. COMPUTATION OF BOUNDS

The use of a linear programming (LP) relaxation is an obvious way to get lower bounds. Unfortunately, the LP bounds would be very poor because the fixed costs are badly represented by the LP relaxation. As noted by [11], in such cases, we can significantly improve the lower bounds in comparison to the linear relaxation lower bounds using the Lagrangean relaxation technique usually couple with a subgradient optimization algorithm [15]. Besides, the use of the Lagrangean relaxation has been extensively applied to important subproblems of the UFNF problem [14, 12].

Dropping the capacity constraints (3) by means of dual variables $w_{ij} \geq 0$, an easily solvable relaxed problem results and the computation of the lower bounds is reduced to solve two polynomial subproblems: (i) a MCNF problem in $\mathbf{x}$ and (ii) a set-selection problem in $\mathbf{y}$. The subproblem in $\mathbf{x}$ is solvable by an $O(|N||A|)$ procedure making use of a shortest *simple* paths algorithm for arbitrary costs [3] since as our initial assumption there are no negative cost circuits[1]. The problem in $\mathbf{y}$ is solvable by an $O(|A|)$ algorithm.

A simple and efficient way of computing upper bounds for model (M) is to take advantage of the relaxed problem solution. The flows obtained after solving the problem in $\mathbf{x}$ are feasible because they satisfy the demand requirements. The only additional work necessary to be done is to compute the cost of these flows using the original costs c_{ij} and adding to the total flows' cost the overhead costs f_{ij} for each arc supporting flows.

3.2. BRANCHING STRATEGY

The simplest way is to choose the first free arc, Figure 2.

```
procedure Choose_First_Free_Arc
        /* search new eligible free arc */
    for all (i,j) ∈ A do
        if (i,j) is still unfixed by the branch-
           and-bound algorithm then
             return (i,j)
        end if
    end for
    return FAIL
end procedure
```

Figure 2: "Naive" Branching Strategy

However, the branching strategy is fundamental in the performance of branch-and-bound algorithms. The use of clever choices coupled with the use of good lower and upper bounds can reduce significantly the number of nodes explicitly examined in the branch-and-bound search tree with consequent reduction in the overall processing time. The optimal solutions of UFNF problems have a very important property that can be very helpful as stated by the following Theorem:

Theorem 1 *If the UFNF problem has an optimum solution then there is an optimum positive flow arc set such that at most one arc enters into each node.*

[1]Note that the shortest *simple* path problem with negative cost circuits is $\mathcal{NP}$-hard [3].

Proof (by contradiction): *Let us suppose that Theorem 1 is not satisfied by any optimal solution and that node m has two entering arcs say (k, m) and (l, m). There must be a set of arcs forming a directed path without cycles from the supply node to node m passing through node k, i.e. using arc (k, m), called P_k. There is also a directed path without cycles using arc (l, m), called P_l. Without loss of generality, let us suppose that*

$$\sum_{(i,j)\in P_k} c_{ij} \leq \sum_{(i,j)\in P_l} c_{ij}.$$

Thus, disabling arc (l, m) and transferring its flow, x_{lm}, from path P_l to path P_k there will be at least the following reduction in the objective function:

$$\left(\sum_{(i,j)\in P_l} c_{ij} - \sum_{(i,j)\in P_k} c_{ij} \right) x_{ij} + f_{lm} \geq 0.$$

The resulting solution is at least as good as the original and satisfies Theorem 1 contradicting our initial assumption. ■

Thus, a possibly much better way is to choose the first free arc found that does not violate Theorem 1, Figure 3.

```
procedure Choose_No_Cycling_Arc
        /* compute set of reached nodes T */
    T ← ∅
    for all i ∈ N such that b_i > 0 do
        T ← T ∪ {i}
    end for
    for all (i, j) ∈ A do
        if (i, j) is already fixed to 1 by the
          branch-and-bound algorithm then
            T ← T ∪ {i} ∪ {j}
        end if
    end for
        /* search new eligible free arc */
    for all (i, j) ∈ A do
        if (i, j) is still unfixed by the branch-
          and-bound algorithm and i ∈ T and
          j ∉ T then
            return (i, j)
        end if
    end for
    return FAIL
end procedure
```

Figure 3: Improved Branching Strategy

The last **for all** in the *Choose_No_Cycling_Arc* procedure is the most expensive operation finishing after $O(|A|)$ iterations in the worst case. Its internal **if** can be made $O(1)$ resulting in an $O(|A|)$ overall worst-case time complexity, surprisingly the same as in the *Choose_First_Free_Arc* procedure.

4. EXPERIMENTAL RESULTS

A preliminary version of the algorithm coded in the *C* programming language was developed and is available upon request. All tests presented were performed using a DECstation 3100 running the operating system ULTRIX V4.2A (Rev. 47).

All test problems came from Euclidean graphs randomly generated using a procedure similar to one presented in [1] that has been extensively applied for creating testing instances [23, 4].

Each test problem was generated as follows. A total of $|N|$ nodes were allocated in a 100×100 square using a uniform probability distribution. The extremities i and j of $|N|-1$ edges were defined also by a uniform distribution ensuring the existence of an undirected spanning tree[2]. Then extremities of additional edges were randomly generated until $|A|/2$ edges were present. In order to convert the above undirected test problems into directed ones, each edge was replaced by 2 opposite arcs. The arc weights Ω_{ij} were defined as the Euclidean distance between the respective extremities.

For all graphs, the nodes were randomly picked up only once and put either in the supply candidate node set S either in the demand node set D creating sets with the required cardinality. All demands were considered unitary. The cardinality of set S was also considered unitary. Graphs with 16 and 32 nodes were used. As much as possible, sparse and dense networks were tested for each number of nodes. For each combination of nodes and arcs, different number of demand nodes were tested. The costs f_{ij} and c_{ij} were derived from the weights Ω_{ij} using the constant factors 1 and 10.

Table 1 presents the results of all computational experiments. The results presented for the first node of the branch-and-bound search tree are the best upper bound, the gap, and the CPU time spent in seconds. The total number of branch-and-bound nodes explored and the CPU time in seconds spent after the first node are presented for both branching strategies. All CPU times reported are the clock time excluding all I/O operations and considering only a single process running on the machine.

For each graph, three instances with different $\frac{f_{ij}}{c_{ij}}$ ratios were considered. The problems with ratio $1:10$ ($f_{ij} = \Omega_{ij}$ and $c_{ij} = 10\Omega_{ij}$) form a class approaching the MCNF problem which is polynomially solvable. On the other hand, the problems with ratio $10:1$ ($f_{ij} = 10\Omega_{ij}$ and $c_{ij} = \Omega_{ij}$) form a class of almost *Steiner* problems which is $\mathcal{NP}$-hard. However, both cases are still $\mathcal{NP}$-hard.

From the column GAP in Table 1, it may be seen that although offering poor lower bounds, mainly in those problems *closer* to *Steiner* problems, the

[2]Such procedure guarantees a connected network and therefore a feasible solution for the problem.

Table 1: Effect of The Branch-and-Bound Strategy

					First Node			Branch-and-Bound			
								Choose_First_Free_Arc		*Choose_No_Cycling_Arc*	
$\|N\|$	$\|A\|$	$\|D\|$	$\frac{f_{ij}}{\Omega_{ij}}$	$\frac{c_{ij}}{\Omega_{ij}}$	SOL[3]	GAP[4]	CPU	Nodes	CPU	Nodes	CPU
16	30	4	1	10	1.0000	1.50	0.20	299	24.00	29	2.40
			1	1	1.0000	12.00	0.20	299	24.00	29	2.40
			10	1	1.0000	44.00	0.22	299	26.00	29	2.50
		8	1	10	1.0000	2.10	0.21	817	69.00	43	3.70
			1	1	1.0000	19.00	0.21	817	69.00	43	3.80
			10	1	1.0000	94.00	0.22	817	75.00	43	4.00
		15	1	10	1.0000	1.90	0.23	4,817	440.00	31	2.90
			1	1	1.0000	18.00	0.24	4,817	440.00	31	2.90
			10	1	1.0000	110.00	0.24	4,817	480.00	31	3.20
	60	4	1	10	1.0000	3.40	0.52	4,143	750.00	189	45.00
			1	1	1.0000	24.00	0.53	6,475	1,200.00	245	61.00
			10	1	1.0000	68.00	0.56	3,389	790.00	375	110.00
		8	1	10	1.0000	3.50	0.55	**	**	3,181	740.00
			1	1	1.0000	25.00	0.52	**	**	4,063	980.00
			10	1	1.0000	110.00	0.57	**	**	11,689	3,100.00
32	62	4	1	10	1.0000	4.20	0.64	196,427	48,000.00	1,923	510.00
			1	1	1.0000	34.00	0.64	**	**	1,923	510.00
			10	1	1.0000	120.00	0.68	**	**	1,923	540.00
		8	1	10	1.0000	3.70	0.65	**	**	3,635	960.00
			1	1	1.0000	33.00	0.63	**	**	3,635	960.00
			10	1	1.0000	170.00	0.67	**	**	3,635	1,000.00
		16	1	10	1.0000	2.70	0.68	**	**	2,071	610.00
			1	1	1.0000	26.00	0.67	**	**	2,071	610.00
			10	1	1.0000	170.00	0.71	**	**	2,071	640.00
		31	1	10	1.0000	2.20	0.74	**	**	63	21.00
			1	1	1.0000	22.00	0.74	**	**	63	21.00
			10	1	1.0000	170.00	0.78	**	**	63	22.00
	124	4	1	10	1.0000	5.40	2.10	**	**	379	330.00
			1	1	1.0024	44.00	2.00	**	**	4,663	4,200.00
			10	1	1.0011	120.00	2.20	**	**	16,817	17,000.00

** Not available (time overflow).

Lagrangean relaxation is a very good heuristic for solving the UFNF problem. Only in two cases the optimum was not reached at the first branch-and-bound node (see the column SOL in Table 1). Of course, the larger and more dense the instances are, the less likely the optimum will be reached in the first branch-and-bound node.

The results presented also shown how the branch-and-bound strategy affects the processing time. The *Choose_No_Cycling_Arc* procedure had a much better performance. The procedure reduces the number of combinations since it disregards all infeasible solutions involving cycles. The worst-case time complexity of the branch-and-bound algorithm under such procedure does not change and it is still $O(2^{|A|})$. However, it is reasonable to expect in practice that the time complexity be lower than this because the problems usually have many arcs forming cycles.

5. FINAL REMARKS

The UFNF problem is a challenging intractable problem ($\mathcal{NP}$-hard) with many applications for the real word. The generic model also encompasses many other special cases with remarkable importance in practice. A formulation for the problem was presented and an algorithm to solve it to optimality was discussed. New criteria for choosing branching variables based on an optimum solution property was introduced. The algorithms were implemented performing very well in practice as the computational experiments have shown.

Some open questions remains. Would it be possible to reduce even more the number of explored nodes using a more complex strategy to choose a free non-cycling arc perhaps taking advantage of the Lagrangean relaxation lower bounds? Future work may include investigations of this question. It may also include the development of reduction tests that eliminate arcs and/or nodes from the original problem in a preprocessing stage, similar to those tests presented in [18] and [10] for the *Steiner* problem in graphs. It is also of interest to investigate how the techniques proposed here can be adapted for solving some special cases of the UFNF problem (*e.g.* the fixed-charge transportation problem [2] and the uncapacitated facility location problem [12]) taking advantage of their particular network structure.

[3] $\text{SOL} = \frac{\text{best upper bound}}{\text{optimal solution}}$

[4] $\text{GAP} = \frac{\text{(best upper bound)}-\text{(best lower bound)}}{\text{best lower bound}} * 100\%$

6. REFERENCES

[1] Y.P. Aneja. An integer linear programming approach to Steiner problem in graphs. *Networks*, 10:167–178, 1980.

[2] R.S. Barr, F. Glover, and D. Klingman. A new optimization method for large scale fixed charge transportation problems. *Operations Research*, 29:448–463, 1981.

[3] M.S. Bazaraa, J.J. Jarvis, and H.D. Sherali. *Linear Programming and Networks Flows*. John Wiley & Sons, New York, 2nd edition, 1990.

[4] J.E. Beasley. An SST-based algorithm for the Steiner problem in graphs. *Networks*, 19:1–16, 1989.

[5] A.V. Cabot and S.S. Erenguc. Some branch-and-bound procedures for fixed-cost transportation problems. *Naval Research Logistics Quarterly*, 31:145–154, 1984.

[6] F.R.B. Cruz, J.A. Almeida, and G.R. Mateus. Análise do problema de planejamento de redes telefônicas de alimentação. In M.A. Silveira and P.M.G. Ferreira, editors, *Anais do 10º Congresso Brasileiro de Automática*, volume 1, pages 572–577, Rio de Janeiro, Brazil, 1994. SBA.

[7] F.R.B. Cruz, H.P.L. Luna, and G.R. Mateus. Uma heurística para o problema de planejamento de redes telefônicas de alimentação. In E.P. Ferreira, editor, *Anais do 9º Congresso Brasileiro de Automática*, pages 443–448, Vitória, Brazil, 1992. SBA.

[8] F.R.B. Cruz, J. MacGregor Smith, and G.R. Mateus. A branch-and-bound algorithm to solve the multi-level network optimization problem. Technical Report RT030/95, Departamento de Ciência da Computação, UFMG, Belo Horizonte, Brazil, 1995. (under review with *Networks*).

[9] F.R.B. Cruz, J. MacGregor Smith, and G.R. Mateus. Solving to optimality the uncapacitated fixed-charge network flow problem. Technical Report RT031/95, Departamento de Ciência da Computação, UFMG, Belo Horizonte, Brazil, 1995. (under review with *Computers & Operations Research*).

[10] C.W. Duin and A. Volgenant. Reduction tests for the Steiner problem in graphs. *Networks*, 19:549–567, 1989.

[11] M.L. Fisher. An application oriented guide to Lagrangean relaxation. *Interfaces*, 15:10–21, 1985.

[12] R.D. Galvão. The use of Lagrangean relaxation in the solution of uncapacitated facility location problems. *Location Science*, 1(1):57–79, 1993.

[13] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[14] B. Gavish. Topological design of computer communication networks - The overall design problem. *European Journal of Operational Research*, 58:149–172, 1992.

[15] M. Held, P. Wolfe, and H.D. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6:62–88, 1974.

[16] D.S. Hochbaum and A. Segev. Analysis of a flow problem with fixed charges. *Networks*, 19:291–312, 1989.

[17] H.P.L. Luna, N. Ziviani, and R.M.B. Cabral. The telephonic switching centre network problem: Formalization and computational experience. *Discrete Applied Mathematics*, 18:199–210, 1987.

[18] N. Maculan, P. Souza, and A.C. Vejar. An approach for the Steiner problem in directed graphs. *Annals of Operations Research*, 33:471–480, 1991.

[19] T.L. Magnanti, P. Mireault, and R.T. Wong. Tailoring Benders decomposition for uncapacitated network design. *Mathematical Programming Study*, 26:112–154, 1986.

[20] G.R. Mateus, F.R.B. Cruz, and H.P.L. Luna. An algorithm for hierarchical network design. *Location Science*, 2(3):149–164, 1994.

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

[22] B. Rothfarb, H. Frank, D.M. Rosembaun, and K. Steiglitz. Optimal design of offshore natural-gas pipeline systems. *Operations Research*, 18:992–1020, 1970.

[23] R.T. Wong. A dual ascent algorithm for the Steiner problem in directed graphs. *Mathematical Programming*, 28:271–287, 1984.