Solving to Optimality the Uncapacitated Fixed-Charge Network Flow Problem

F.R.B. Cruz [a,1], J. MacGregor Smith [b,2], and G.R. Mateus [c,3,4]

[a] *Programa de Pós-Graduação em Engenharia Elétrica, Pontifícia Universidade Católica de Minas Gerais, 30535-610 - Belo Horizonte - MG, Brazil*

[b] *Department of Mechanical and Industrial Engineering, The University of Massachusetts, Amherst, MA 01003, USA*

[c] *Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, 31270-010 - Belo Horizonte - MG, Brazil*

Scope and Purpose

Given a directed network, the uncapacitated fixed-charge network flow problem is to find a minimum-cost arc combination that provides flows from the supply nodes to the demand nodes. Associated with all arcs are two costs: 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. In this paper, we use the well-known branch-and-bound algorithm to exactly solve the problem. We develop a computationally efficient branching strategy and a reduction technique based on Lagrangean Relaxation. We also propose a simplification of the branch-and-bound algorithm. As a result of these strategic improvements, we achieve a significant speed-up in the solution process.

Abstract

We present the uncapacitated fixed-charge network flow problem and two mathematical programming formulations. We use an exact approach to solve the problem, the well-known branch-and-bound algorithm. We derive bounds for the algorithm using Lagrangean Relaxation and also propose an efficient branching strategy which is based on an important property of the optimal solution. We also use a Lagrangean Relaxation of the problem to develop a new reduction test. The practical efficiency of all the procedures is demonstrated through a comprehensive set of computational experiments.

Preprint submitted to Elsevier *14 August 1997*

[1] F.R.B. Cruz received his Doctor degree from *Universidade Federal de Minas Gerais*, Belo Horizonte, Brazil. His research areas include network optimization and operations research. During this work, he was a Visiting Research Scholar in the Department of Mechanical and Industrial Engineering at the University of Massachusetts at Amherst, USA, supported by the *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq), Brazil, under grant CNPq 201046/94-6. Currently, he is a Visiting Associate Professor in the *Programa de Pós-Graduação em Engenharia Elétrica* at *Pontifícia Universidade Católica de Minas Gerais*, Belo Horizonte, Brazil, supported by FAPEMIG, Brazil, under grant TEC-1927/96. E-mail: fcruz@dcc.ufmg.br.

[2] J. MacGregor Smith received his Ph.D. degree from University of Illinois at Urbana-Champaign, Illinois, USA. His research areas include combinatorial optimization, mathematical programming, and applied operations research. Currently, he is a Professor in the Department of Mechanical and Industrial Engineering at the University of Massachusetts at Amherst, USA. E-mail: jmsmith@ecs.umass.edu.

[3] G.R. Mateus received his Doctor degree in 1986 from *Universidade Federal do Rio de Janeiro*, Rio de Janeiro, Brazil. His research areas include network optimization and operations research. Currently, he is a Full Professor in the *Departamento de Ciência da Computação*, at *Universidade Federal de Minas Gerais*, Belo Horizonte, Brazil. E-mail: mateus@dcc.ufmg.br.

[4] To whom all correspondences should be addressed. Address: Caixa Postal 702, Departamento de Ciência da Computação - UFMG, 30123-970 - Belo Horizonte - MG, Brazil.

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.

Figure 1

This is clearly an $\mathcal{NP}$-hard optimization problem since it generalizes the *Steiner* problem in graphs, among others, and its $\mathcal{NP}$-hardness was shown in [1]. This generic model has applications for problems of distribution, transportation and communication. It is also useful for certain routing 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.

Numerous exact and approximate solutions for the UFNF problems and their special cases have been published previously. In [2], an analysis of offshore natural-gas systems was done with the cost model being simplified including only fixed costs. In [3], 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 [4] and [5]. 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 technique was developed. In [6], [7] and [8], some special cases were solved using a branch-and-bound algorithm with fractional cutting-planes. In [9], they introduced a new family of dicut collection inequalities for the general model using multicommodity extended formulations. Previously in [10], we have developed *ADD* and *DROP* heuristic approaches and in [11], we have implemented simplified branch-and-bound algorithms to solve the UFNF problem.

In this paper, we are tackling the problem exactly by means of a branch-and-bound algorithm. This enumerative approach is known to be computationally inefficient because of its combinatorial explosive behavior. Although it is often only acceptable for small problems instances, this approach is the one of the

most preferred to solve exactly $\mathcal{NP}$-hard problems. However, it is also know that the use of efficient branching strategies and reduction techniques can significantly enlarge the size of the manageable instances. In this sense, the focus of this paper is to develop new criteria to choose the branching variables, to derive a new reduction test and to propose a simplification in the branch-and-bound algorithm. Of course, we are aware that there always will be cases where an exact approach may be impossible no matter what computer systems or algorithms are utilized.

The paper is outlined as follows. In Section 2, the UFNF problem is presented in a mixed-integer mathematical programming formulation. An alternative formulation is derived. In Section 3, we shall discuss the solution methods we are using, the branching strategy we propose and the reduction algorithm we develop. All algorithms were implemented and our experimental results are reported in Section 4. Section 5 closes this work with final remarks, open questions, and some possible extensions.

2 Problem Formulation

A typical mixed-integer mathematical programming formulation for the problem, over digraph $\mathcal{D} = (N, A)$ is:

(M)**:**

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

s.t.:

$$\sum_{j\in\delta^-(i)} x_{ji} - \sum_{j\in\delta^+(i)} x_{ij} = \begin{cases} -\sum_{k\in D} d_k, & i = s, \\ 0, & \forall i \in T, \\ d_i, & \forall i \in D, \end{cases} \tag{2}$$

$$x_{ij} \leq \left(\sum_{k\in D} d_k\right) y_{ij}, \quad \forall (i,j) \in A, \tag{3}$$

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

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

where N is the set of nodes, A is the set of arcs, $\delta^+(i) = \{j|(i,j) \in A\}$,

$\delta^-(i) = \{j|(j,i) \in A\}$, $s \in N$ is the source node, $T \subseteq N$ is the set of transshipment or *Steiner* nodes, $D \subseteq N$ is the set of demand nodes, d_i is the demand of node i, f_{ij} is the fixed cost of having flow on arc (i,j), and c_{ij} is the variable cost per unit of flow on arc (i,j). It is noticeable that the only difference between the UFNF problem and the uncapacitated 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 $c_{ij}x_{ij} + f_{ij}$. Constraints 3 ensure that characteristic and that simple difference transforms the polynomially solvable MCNF problem into the $\mathcal{NP}$-hard UFNF problem.

We now postulate an assumption about the UFNF problem. 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}.

During this paper, we also consider an alternative and equivalent formulation for the UFNF problem. Therefore, let us define $K_0 \subseteq A$ as the set of arcs that have been rejected, $y_{ij} = 0$, $K_1 \subseteq A$ as the set of arcs that have been selected, $y_{ij} = 1$, and $K = A \setminus K_0 \setminus K_1$ as the set of arcs in an undefined status. The UFNF problem can be alternatively represented by the following formulation:

(M')**:**

$$\min \sum_{(i,j)\in A} \left(c_{ij}x_{ij} + f_{ij}y_{ij}\right), \tag{6}$$

s.t.:

$$\sum_{j\in\delta^-(i)} x_{ji} - \sum_{j\in\delta^+(i)} x_{ij} = \begin{cases} -\sum_{k\in D} d_k, & i = s, \\ 0, & \forall i \in T, \\ d_i, & \forall i \in D, \end{cases} \tag{7}$$

$$x_{ij} \leq \left(\sum_{k\in D} d_k\right) y_{ij}, \quad \forall (i,j) \in K, \tag{8}$$

$$x_{ij} \geq 0, \quad \forall (i,j) \in K, \tag{9}$$

$$y_{ij} \in \{0,1\}, \quad \forall (i,j) \in K, \tag{10}$$

$$x_{ij} \leq \sum_{k\in D} d_k, \quad \forall (i,j) \in K_1, \tag{11}$$

$$x_{ij} \geq 0, \quad \forall (i,j) \in K_1, \tag{12}$$
$$y_{ij} = 1, \quad \forall (i,j) \in K_1, \tag{13}$$

$$x_{ij} = 0, \quad \forall (i,j) \in K_0, \tag{14}$$
$$y_{ij} = 0, \quad \forall (i,j) \in K_0. \tag{15}$$

The above formulation is more convenient than the previous. First, because it is immediate that the model (M') represents exactly the same problem as model (M) if $K_1 = K_0 = \emptyset$. The model (M') also can represent each of the subproblems generated by the branch-and-bound algorithm in the search tree. Finally, it can even represent the problem after the application of our reduction test.

3 Solution Method

We need to employ an enumerative scheme if we intend to solve exactly this $\mathcal{NP}$-hard optimization problem. Such an enumeration algorithm is frequently called branch-and-bound or implicit enumeration. We use the recursive branch-and-bound algorithm presented below in which the depth-first search strategy is in use.

algorithm Solve(M')
/* bounding */
compute lower bound L
compute upper bound U *and update* U_{BEST}
GAP $\leftarrow \frac{U-L}{U}$
/* branching */
if $L > U_{\text{BEST}}$ **then**
write 'Infeasible node reached.'
else if GAP $\leq \varepsilon$ **then**
write 'Optimum reached.'
else if $K = \emptyset$ **then**
write 'Leaf reached.'
else
$(i, j) \leftarrow$ Chosen_Arc_$*$
$K \leftarrow K \setminus (i, j)$; $K_1 \leftarrow K_1 \cup (i, j)$
Solve(M')
$K_1 \leftarrow K_1 \setminus (i, j)$; $K_0 \leftarrow K_0 \cup (i, j)$
Solve(M')
$K_0 \leftarrow K_0 \setminus (i, j)$; $K \leftarrow K \cup (i, j)$
end if
end if
end if
end algorithm

Now we present the flowchart version of the algorithm. In the description, Γ is a collection of problems $\{(M')^k\}$, each of which is of the form $Z^k_{M'} = \min\{\sum_{(i,j)\in A}(c_{ij}x_{ij} + f_{ij}y_{ij})$ **s.t.:** (7)–(15), $K_0 = K^k_0 \subseteq A, K_1 = K^k_1 \subseteq A, K = K^k = A \setminus K^k_0 \setminus K^k_1\}$. Associated with each problem in Γ are a lower bound $L^k \leq Z^k_{M'}$ and an upper bound $U^k \geq Z^k_{M'}$. The flowchart of our algorithm we use is depicted in Figure 2 which mirrors the classic branch-and-bound algorithm presented in [12]. To transform the recursive algorithm above into this non-recursive flowchart, we applied the well known procedure, described in [13]. The policy we use to select a problem $(M')^i$ from the list Γ is *last-in-first-out* which yields a depth-first search strategy. An important property of this search is that the maximum length of the list Γ is kept as short as possible with memory economy. Further details on this strategy as well as others can be found in [12]. If a solution does exist, the algorithm in Figure 2 always solves it because it can enumerate implicitly all possibilities. For further details, *i.e.* a proof of correctness, the reader is encouraged to check on [14]. The algorithm in its virgin form is computationally inefficient because of its exponential worst-case time complexity, $O(2^{|A|})$, and it is only acceptable for small sized problem instances. However, if we provide tighter lower and upper bounds, an efficient branching strategy, and effective reduction techniques we can significantly enlarge the size of manageable problem instances. We now describe how to compute the tighter bounds L^k and U^k required by the algorithm along with

a new strategy to select an arc (i, j) such that $(i, j) \in K^k$, see Figure 2, and finally a new reduction technique for the problem.

Figure 2

3.1 Lower Bounds

The use of a linear programming (LP) relaxation is an obvious way to get lower bounds for the problem but we are going to use a Lagrangean relaxation approach. As noted in [15], applying Lagrangean relaxation to problems such as the UFNF problem, the lower bounds obtained are no better than the LP relaxation lower bounds. However, as we shall see shortly, it will be possible to derive a heuristic based on Lagrangean relaxation that will provide very tight upper bounds. Besides, the use of the Lagrangean relaxation has been extensively applied to important subproblems of the UFNF problem with successful results. Lagrangean relaxation has been used to solve the distribution system design problems, [2], the uncapacitated location problem, [16,17], and telecommunication network oriented problems, [18,19].

Dropping the capacity constraints (8) by means of dual variables $w_{ij} \geq 0$, $\forall (i, j) \in K$, the following Lagrangean function results:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \sum_{(i,j)\in A} (c_{ij}x_{ij} + f_{ij}y_{ij}) + \sum_{(i,j)\in K} w_{ij} \left[x_{ij} - \left(\sum_{k\in D} d_k \right) y_{ij} \right]. \quad (16)$$

Consequently, the Lagrangean relaxation of model (M') may be written:

$(LR_{\mathbf{w}})$**:**

$$L(\mathbf{w}) = \min\{\mathcal{L}(\mathbf{x}, \mathbf{y}; \mathbf{w}) \textbf{ s.t.: } (7), (9)\text{–}(15)\mathbf{w} \geq 0\}. \quad (17)$$

For any feasible Lagrangean multiplier vector, $\mathbf{w} \geq 0$, the Lagrangean relaxation optimal solution is a lower bound for the original problem because the quantity $\sum_{(i,j)\in K} w_{ij}[{x_{ij}}^* - (\sum_{k\in D} d_k){y_{ij}}^*]$ is always non-positive, [15], supposing that $L(\mathbf{w}) = \mathcal{L}(\mathbf{x}^*, \mathbf{y}^*; \mathbf{w})$. Thus, the computation of the lower bounds reduces to solving two easy (polynomial) subproblems, a MCNF problem in $\mathbf{x}$ and a set-selection problem in $\mathbf{y}$:

(LR_1)**:**

$$\min \sum_{(i,j)\in A} C_{ij}x_{ij}, \tag{18}$$

s.t.:

$$\sum_{j\in\delta^-(i)} x_{ji} - \sum_{j\in\delta^+(i)} x_{ij} = \begin{cases} -\sum_{k\in D} d_k, & i = s, \\ 0, & \forall i \in T, \\ d_i, & \forall i \in D, \end{cases} \tag{19}$$

$$x_{ij} \geq 0, \quad \forall (i,j) \in K, \tag{20}$$

$$x_{ij} \leq \sum_{k\in D} d_k, \quad \forall (i,j) \in K_1, \tag{21}$$

$$x_{ij} \geq 0, \quad \forall (i,j) \in K_1, \tag{22}$$

$$x_{ij} = 0, \quad \forall (i,j) \in K_0, \tag{23}$$

where

$$C_{ij} = \begin{cases} c_{ij} + w_{ij}, & \forall (i,j) \in K, \\ c_{ij}, & \forall (i,j) \in A \setminus K. \end{cases} \tag{24}$$

(LR_2)**:**

$$\min \sum_{(i,j)\in A} F_{ij}y_{ij}, \tag{25}$$

s.t.:

$$y_{ij} \in \{0,1\}, \forall (i,j) \in K, \tag{26}$$

$$y_{ij} = 1, \forall (i,j) \in K_1, \tag{27}$$

$$y_{ij} = 0, \forall (i,j) \in K_0, \tag{28}$$

where

$$F_{ij} = \begin{cases} f_{ij} - w_{ij}(\sum_{k\in D} d_k), & \forall (i,j) \in K, \\ f_{ij}, & \forall (i,j) \in A \setminus K. \end{cases} \tag{29}$$

The subproblem in $\mathbf{x}$ is solvable with polynomial time complexity, $O(|N||A|)$, as shown by [20], by use of a shortest *simple* paths algorithm for arbitrary costs developed by [21], since, as our initial assumption, there are no negative cost circuits [1] . The problem in $\mathbf{y}$ is also solvable with polynomial complexity, $O(|A|)$, as all we need to do is to set to 1 all those y_{ij}'s for which $F_{ij} < 0$.

The traditional way of improving the lower bounds obtained by Lagrangean relaxation procedures is to apply a subgradient optimization algorithm, *e.g.* the method was outlined in [22]. In this paper, we initialize the Lagrangean multipliers $\mathbf{w}$ with $\mathbf{0}$ only in the first call of the algorithm. All subsequent calls should not reinitialize them because there is no reason the last previously used vector would not be used as the starting point $\mathbf{w}_0$.

3.2 Upper Bounds

We propose a simple and efficient way of getting upper bounds. The flows obtained after solving the (LR_1) are feasible in the primal problem because they are satisfying 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 it the overhead costs f_{ij} for each arc that supports flows.

3.3 Branching Strategies

The strategy of choosing the branching variable is fundamental to the performance of the branch-and-bound algorithm. The use of clever choices of branching variables coupled with the use of good lower and upper bounds can reduce significantly the number of nodes explicitly examined in the search tree with consequent reduction in the processing time. We are using two different strategies in this work. The simplest way is to choose the first free arc found accordingly to the procedure shown below.

```
procedure Chosen_Arc_First_Free
    for all (i, j) ∈ A do
        if (i, j) ∈ K then
            return (i, j)
        end if
    end for
    return FAIL
end procedure
```

[1] The shortest *simple* path problem with negative cost circuits is $\mathcal{NP}$-hard, as shown by [20].

In Figure 3, we present the flowchart for the algorithm which has a worst-case time complexity $O(|N|^2)$, as the command "**if**$(i,j) \in K$**then**" can be made $O(1)$.

Figure 3

However, the optimal solution of the UFNF problem as formulated in model (M) has an important property that will be very helpful in improving the performance of the branch-and-bound algorithms. The property is stated by the following Theorem:

Theorem 1 *If the UFNF problem, model (M), has a finite optimum then there is an optimum positive flow arc set such that at most one arc enters into each node.*

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, by 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, there is a possibly a much better way to choose the first free arc found that does not violate Theorem 1 and this is described in the following procedure:

```
procedure Chosen_Arc_No_Cycling

        /* compute set of reached nodes R */
    R ← {s}
    for all (i, j) ∈ A do
        if (i, j) ∈ K_1 then
            R ← R ∪ {i} ∪ {j}
        end if
    end for

        /* search new eligible free arc */
    for all (i, j) ∈ A do
        if (i, j) ∈ K and i ∈ R and j ∉ R then
            return (i, j)
        end if
    end for
    return FAIL
end procedure
```

The flowchart of the improved branching strategy is illustrated in Figure 4. The last command "**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 command "**if** $(i, j) \in K$ **and** $i \in R$ **and** $j \notin R$ **then**" 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. Thus, the latter strategy will perform *not* worse than the former.

Figure 4

3.4 Reducing the Problem

We now present our algorithm to reduce the problem. The main algorithm has to be modified. When used, the reduction test must be applied just before each arc selection.

The new lower bound that would result from forcing the arcs *in* or *out of* the solution can be easily estimated from the Lagrangean relaxation. If the lower bound resulting from imposing some condition to the Lagrangean relaxation is greater than the best upper bound, then the condition in consideration cannot be satisfied in the optimum. This idea is inspired by the reduction procedures developed in [23] to solve the p-median problem, with very good results in practice. In [23], some terms of the corresponding Lagrangean function have

been used to estimate the increment in the lower bound under the imposed condition. We propose the reduction procedure below, that uses estimated lower bounds computed by means of a complete resolution of the Lagrangean dual problem, $L(\mathbf{w})$, but without subgradient optimizations.

```
procedure Reduce(M')
        /* initialize set of arcs recently fixed */
    F ← ∅
        /* proceed with reduction */
    for all (i, j) ∈ A do
        if (i, j) ∈ K then
            K ← K \ (i, j)
            K_1 ← K_1 ∪ (i, j)
            if L(w) > U_BEST then
                K_1 ← K_1 \ (i, j)
                K_0 ← K_0 ∪ (i, j)
                F ← F ∪ (i, j)
            else
                K_1 ← K_1 \ (i, j)
                K_0 ← K_0 ∪ (i, j)
                if L(w) > U_BEST then
                    K_0 ← K_0 \ (i, j)
                    K_1 ← K_1 ∪ (i, j)
                    F ← F ∪ (i, j)
                else
                    K_0 ← K_0 \ (i, j)
                    K ← K ∪ (i, j)
                end if
            end if
        end if
    end for
end procedure
```

The flowchart of the reduction test procedure is depicted in Figure 5. The reduction procedure stops after the examination of each arc exactly once, $O(|A|)$. Some variations are immediate, *e.g.* passing through each arc twice, and so on. Each iteration involves at most two lower bound calculations which are each $O(|N||A|)$. We remind the reader that no subgradient optimization is performed here. Additionally, at most four set insertions and deletions are involved which are $O(|A|)$, but the computation of function $L(\mathbf{w})$ dominates. Therefore, the procedure will run with a worst-case time complexity $O(|N||A|^2)$.

Figure 5

As a final remark, the $O(|A|^2)$ procedure presented below must be activated in the back-tracking stage in order to reset the problem (M') to its status just before the reduction. This procedure should be applied after the second recursive call in the main algorithm. Note that in the flowchart version of the main algorithm, Figure 2, there is no need to apply such a procedure. Once a problem $(M')^i$ has created its children, $(M')^{2i+1}$ and $(M')^{2i+2}$, such a problem will not be used again and it does not have to be reset.

```
procedure Unreduce(M')
    for all (i,j) ∈ A do
        if (i,j) ∈ F then
            K ← K ∪ (i,j)
            if (i,j) ∈ K0 then
                K0 ← K0 \ (i,j)
            else
                K1 ← K1 \ (i,j)
            end if
        end if
    end for
end procedure
```

4 Experimental Results

All tests presented here were performed in a DECstation 3100 running the operating system ULTRIX V4.2A (Rev. 47) and a preliminary version of the algorithm coded in the *C* programming language available from the authors upon request. The parameter ε in the branch-and-bound algorithm was set to 10^{-6} which is close to the precision offered by the compiler used when representing real numbers under floating point data type.

All test problems came from Euclidean graphs randomly generated using the scheme outlined in [24] and extensively used by others, *e.g.* [25], [26], and [27], for the *Steiner* problem in graphs. The arc costs Ω_{ij} were defined as the Euclidean distance between the extremities i and j. The problems presented in Table 1 were generated by a code in *C* developed by us. Problems presented in Table 2 are those solved in [26] and [27], made available to us by Beasley [28].

The demand was considered unitary for all demand nodes. The costs f_{ij} and c_{ij} were derived from the distances Ω_{ij} using the constant factors 1 and 10. For each graph, three different instances with different f_{ij}/c_{ij} ratios were considered. The problems with ratio 1 : 10 form a class of problems "closer" to

the MCNF problem which is polynomially solvable. On the other hand, the problems with ratio 10 : 1 form a class of problems "closer" to the *Steiner* problem which is $\mathcal{NP}$-hard. However, both classes are still $\mathcal{NP}$-hard. For each test instance, we present the following results for the initial node in the search tree: the best upper bound, the gap, and the CPU time in seconds excluding all I/O operations and considering that only a single process was running on the machine. For the whole tree search, unless we had time overflow, 20,000 seconds, we present the optimal solution, the number of nodes and the CPU time in seconds spent until the optimum was reached for four combinations: (i) using the branching strategy of Figure 3, (ii) using the improved branching strategy of Figure 4, (iii) combining the latter strategy with the reduction technique presented in Figure 5, and (iv) using the third combination with the exclusion of the subgradient optimization algorithm after the first node on the search tree.

Table 1

Table 2

From the results presented in Table 1 and in Table 2, it may be seen that although offering poor lower bounds, mainly in those problems "closer" to *Steiner* problems, the Lagrangean relaxation developed here is a very good heuristic for solving the UFNF problem. Only in one problem in Table 2, was the optimum not reached at the first node of the search tree. Of course, the larger and more dense the instances are, the less likely the optimum will be reached in the first node.

The results also indicate how positively the improved branching strategy affects the CPU time. The reduction in the processing time caused by the procedure observed here will also occur in larger networks. The procedure drops the number of combinations as it disregards all infeasible solutions involving cycles. The impact of the the reduction algorithm is remarkable. In sparse instances, the reduction algorithm kept the number of explored nodes surprisingly low. The algorithm was able to solve quickly dense networks if the number of demand nodes was low and the problems were closer to the MCNF problems. The problems closer to *Steiner* problems were really much harder.

An interesting issue emerged during this work was the effect of the subsequent subgradient optimizations. It was noted in practice that small increments were obtained in the lower bounds by the subgradient optimization algorithm after the tree search first node if the Lagrangean multipliers were not reinitialized. After reducing and fixing variables, the remaining Lagrangean multipliers are still feasible. Besides, the multipliers are already very good and the lower bounds obtainable from them are also very good. Therefore, we decided to ap-

Table 1. Computational Results for Random Networks ($|N| = 16$ and $|N| = 32$)

					First node			Tree search								
									Choose_First_Free_Arc		*Choose_No_Cycling_Arc*		Plus Reduction		Plus Simplification	
$\|N\|$	$\|A\|$	$\|D\|$	$\frac{f_{ij}}{\Omega_{ij}}$	$\frac{c_{ij}}{\Omega_{ij}}$	U_{BEST}	GAP(%)	CPU(s)	U_{OPT}	Nodes	CPU(s)	Nodes	CPU(s)	Nodes	CPU(s)	Nodes	CPU(s)
16	30	4	1	10	5,972	1.50	0.20	5,972	301	23.00	35	2.80	1	0.03	1	0.03
			1	1	806	11.00	0.20	806	301	23.00	35	2.70	1	0.03	1	0.03
			10	1	2,894	31.00	0.22	2,894	301	25.00	35	3.00	1	0.03	1	0.03
		8	1	10	12,250	2.10	0.21	12,250	819	68.00	35	2.90	1	0.04	1	0.04
			1	1	1,585	16.00	0.21	1,585	819	68.00	35	2.90	1	0.04	1	0.04
			10	1	5,185	49.00	0.22	5,185	819	73.00	35	3.20	1	0.04	1	0.04
	60	4	1	10	4,066	4.20	0.52	4,066	9,445	1,900.00	85	19.00	1	0.12	1	0.12
			1	1	646	26.00	0.53	646	18,941	3,700.00	259	61.00	31	16.00	57	5.10
			10	1	2,400	45.00	0.59	2,400	**	**	557	150.00	43	23.00	57	5.50
32	62	4	1	10	7,645	4.00	0.64	7,645	**	**	905	240.00	1	0.18	1	0.18
			1	1	1,201	25.00	0.65	1,201	**	**	905	230.00	1	0.18	1	0.18
			10	1	5,566	55.00	0.72	5,566	**	**	905	250.00	1	0.17	1	0.18
		8	1	10	12,349	3.60	0.66	12,349	**	**	4,261	1,100.00	1	0.19	1	0.20
			1	1	1,765	25.00	0.64	1,765	**	**	4,261	1,100.00	1	0.19	1	0.20
			10	1	7,066	63.00	0.67	7,066	**	**	4,261	1,100.00	1	0.20	1	0.20
		16	1	10	30,093	2.60	0.69	30,093	**	**	2,255	640.00	1	0.21	1	0.21
			1	1	3,885	20.00	0.69	3,885	**	**	2,255	641.00	1	0.21	1	0.21
			10	1	12,642	63.00	0.73	12,642	**	**	2,255	670.00	1	0.21	1	0.21
		31	1	10	62,499	2.20	0.80	62,499	**	**	61	22.00	1	0.24	1	0.24
			1	1	7,644	18.00	0.77	7,644	**	**	65	22.00	1	0.24	1	0.25
			10	1	21,585	63.00	0.82	21,585	**	**	65	23.00	1	0.25	1	0.25
	124	4	1	10	6,891	3.20	1.90	6,891	**	**	2,255	2,400.00	9	18.00	9	4.00
			1	1	1,026	21.00	1.90	1,016	**	**	**	**	165	340.00	269	110.00
			10	1	3,878	41.00	2.30	3,878	**	**	**	**	597	1,200.00	969	360.00
		8	1	10	14,484	3.00	2.00	14,484	**	**	**	**	15	35.00	15	9.30
			1	1	1,986	21.00	2.00	**	**	**	**	**	**	**	**	**
			10	1	6,845	54.00	2.10	**	**	**	**	**	**	**	**	**
		16	1	10	25,023	2.30	2.10	25,023	**	**	**	**	19	46.00	19	13.00
			1	1	3,147	18.00	2.70	**	**	**	**	**	**	**	**	**
			10	1	8,964	56.00	2.20	**	**	**	**	**	**	**	**	**
	248	4	1	10	2,821	5.90	6.40	2,821	**	**	**	**	63	420.00	63	83.00
			1	1	468	31.00	6.40	468	**	**	**	**	337	2,100.00	367	360.00
			10	1	1,926	53.00	6.80	**	**	**	**	**	**	**	**	**
		8	1	10	5,200	6.80	6.40	5,200	**	**	**	**	679	4,900.00	679	1,100.00
			1	1	861	37.00	6.60	**	**	**	**	**	**	**	**	**
			10	1	3,696	71.00	6.60	**	**	**	**	**	**	**	**	**

**Not available (time overflow)

Table 2. Computational Results for Beasley's Networks [28] ($\frac{f_{ij}}{\Omega_{ij}} = 1$ and $\frac{c_{ij}}{\Omega_{ij}} = 10$)

							Tree search								
				First node				*Choose_First_Free_Arc*		*Choose_No_Cycling_Arc*		Plus Reduction		Plus Simplification	
Problem	$\|N\|$	$\|A\|$	$\|D\|$	U_{BEST}	GAP(%)	CPU(s)	U_{OPT}	Nodes	CPU(s)	Nodes	CPU(s)	Nodes	CPU(s)	Nodes	CPU(s)
B-1	50	126	8	1,222	5.60	2.30	1,222	**	**	**	**	1	1.10	1	1.10
2			12	2,520	2.80	2.30	2,520	**	**	**	**	1	3.20	1	2.70
3			24	5,017	3.10	2.40	5,012	**	**	**	**	495	1,300.00	281	170.00
4		200	8	1,237	5.10	4.90	**	**	**	**	**	**	**	**	**
5			12	1,095	5.20	4.80	1,095	**	**	**	**	1,676	8,800.00	1,799	1,800.00
6			24	3,208	4.20	5.90	**	**	**	**	**	**	**	**	**
7	75	188	12	2,943	3.40	4.80	2,943	**	**	**	**	1	4.40	1	4.50
8			18	2,657	3.90	5.00	2,657	**	**	**	**	9	55.00	7	16.00
9			37	5,874	3.70	5.20	5,874	**	**	**	**	5	31.00	5	12.00
10		300	12	2,053	5.20	11.00	**	**	**	**	**	**	**	**	**
11			18	3,987	2.70	11.00	3,987	**	**	**	**	67	1,100.00	61	297.00
12			37	6,948	3.00	12.00	**	**	**	**	**	**	**	**	**
13	100	250	16	4,432	3.70	9.30	4,432	**	**	**	**	**	**	6,517	19,000.00
14			24	9,117	2.60	9.50	**	**	**	**	**	**	**	**	**
15			49	11,383	2.90	9.40	11,383	**	**	**	**	**	**	1,923	4,800.00
16		400	16	3,942	3.50	24.00	**	**	**	**	**	**	**	**	**
17			24	5,193	2.40	24.00	**	**	**	**	**	**	**	**	**
18			49	6,360	4.20	24.00	**	**	**	**	**	**	**	**	**

**Not available (time overflow)

ply the subgradient optimization algorithm only once. The results are shown in the last two columns of Table 1 and Table 2. In some instances, if the subgradient optimization algorithm is used only once, it was possible to observe small increments in the number of explored nodes. However, in all cases the decrement in the computation time was considerable.

5 Summary and Conclusions

Exact solutions for the UFNF problem are very difficult since the problem is $\mathcal{NP}$-hard. Branch-and-bound algorithms are some of the most effective means for solving these problems, however, they are often acceptable only for small problem instances because of their exponential running time behavior. We have proposed a set of supporting algorithmic enhancements that reduce the computation time consequently enlarging the size of manageable problem instances. We have shown by experiments that our new branching criteria and our reduction techniques really make faster solutions possible. Additionally, the ideas behind these supporting procedures are applicable to other similar problems and subproblems of the UFNF problem. Indeed, we have been able to extend these ideas to a multi-level network design (MLND) problem. The MLND problem was introduced in [29] and it is a combination of UFNF problems and uncapacitated discrete location problems and we have obtained encouraging results.

Nevertheless, some open questions remain. Would it be possible to develop a strategy to choose one arc among all free non-cycling arcs instead of choosing the first available? Would the reduction test be more effective if each arc were examined more than once? Future work may include investigations of these questions. Also we may include the development of additional reduction tests that eliminate arcs and nodes or both from the original problem in a preprocessing stage, similar to those procedures developed in [30] and [27] for the *Steiner* problem in graphs. It is also of interest to investigate how the ideas developed here could be adapted to algorithms specialized in particular cases of the UFNF problem, *e.g.* the fixed charge transportation problem studied in [6], the *Steiner* problems in graphs, in [26], or the uncapacitated facility location problem, in [31].

References

[1] 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.

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

[3] H.P.L. Luna, N. Ziviani, and R.M.B. Cabral. The telephonic switching centre network problem: Formalization and computational experience. *Disc. Appl. Math.*, 18:199–210, 1987.

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

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

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

[7] A.V. Cabot and S.S. Erenguc. Some branch-and-bound procedures for fixed-cost transportation problems. *Nav. Res. Logist. Q.*, 31:145–154, 1984.

[8] U. Suhl. Solving large scale mixed integer programs with fixed charge variables. *Math. Programming*, 32:165–182, 1985.

[9] R. L. Rardin and L. A. Wolsey. Valid inequalities and projecting the multicommodity extended formulation for uncapacitated fixed charge network flow problems. *Eur. J. Opl Res.*, 71:95–109, 1993.

[10] 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.

[11] 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*).

[12] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*, chapter II.4 - General Algorithms, pages 349–382. John Wiley & Sons, New York, 1988.

[13] E. Horowitz and S. Sahni. *Fundamentals of Computer Algorithms*, chapter 1 - Introduction, pages 20–24. Computer Science Press, Inc., Potomac, MD, 1978.

[14] E. Horowitz and S. Sahni. *Fundamentals of Computer Algorithms*, chapter 8 - Branch-and-Bound, pages 370–421. Computer Science Press, Inc., Potomac, MD, 1978.

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

[16] R.D. Galvão and L.A. Raggi. A method for solving to optimality uncapacitated location problems. *Annals of Opns Res.*, 18:225–244, 1989.

[17] G.R. Mateus and J.C.P. Carvalho. O problema de localização não capacitado: Modelos e algoritmos. *Investigación Operativa*, 2:297–317, 1992.

[18] B. Gavish. Topological design of telecommunication networks - Local access design methods. *Annals of Opns Res.*, 33:17–71, 1991.

[19] B. Gavish. Topological design of computer communication networks - The overall design problem. *Eur. J. Opl Res.*, 58:149–172, 1992.

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

[21] F. Glover, D. Klingman, and N. Phillips. A new polynomially bounded shortest path algorithm. *Opns Res.*, 33(1):65–73, 1985.

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

[23] N. Christofides and J.E. Beasley. A tree search algorithm for the p-median problem. *Eur. J. Opl Res.*, 10:196–204, 1982.

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

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

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

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

[28] J.E. Beasley. OR-Library: Distributing test problems by electronic mail. *Journal of the Opl Res. Society*, 41(11):1069–1072, 1990.

[29] F.R.B. Cruz, J. MacGregor Smith, and G.R. Mateus. Algorithms for the multi-level network optimization problem. Technical Report RT007/96, Departamento de Ciência da Computação, UFMG, Belo Horizonte, Brazil, 1996. (under review with *Eur. J. Opl Res.*).

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

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

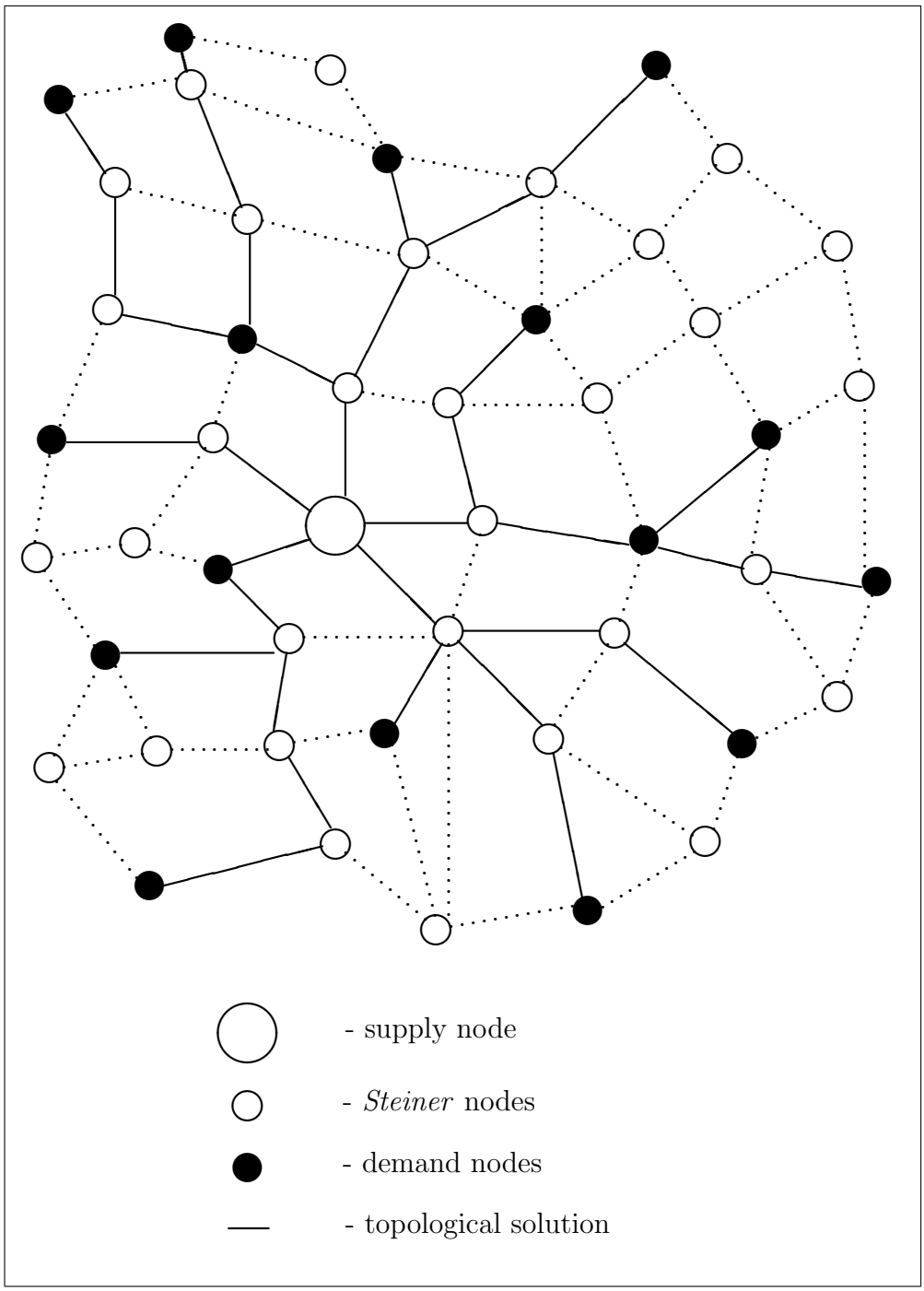

Fig. 1. The Uncapacitated Fixed-charge Network Flow Problem

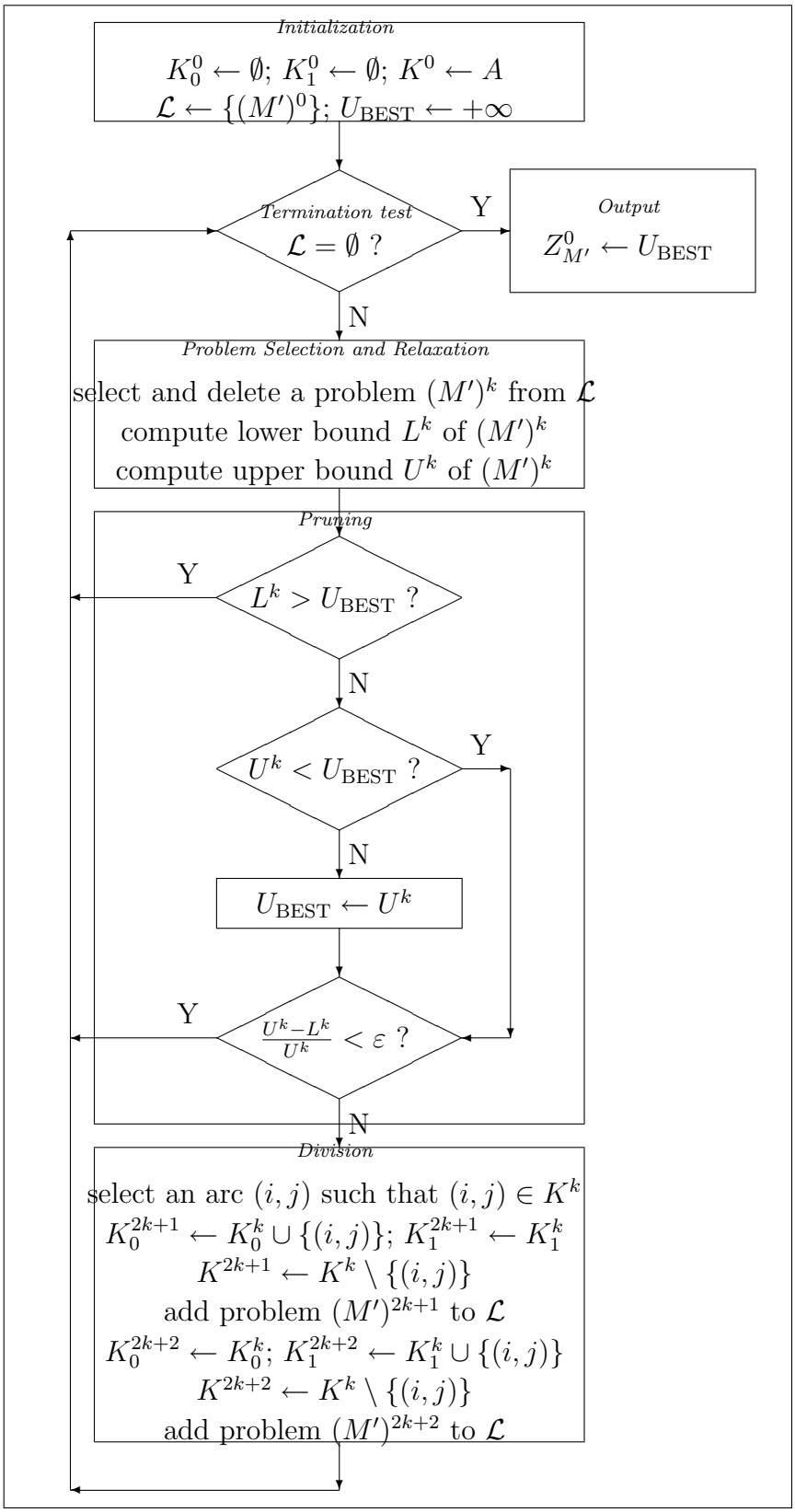

Fig. 2. Flowchart of the Branch-and-Bound Algorithm

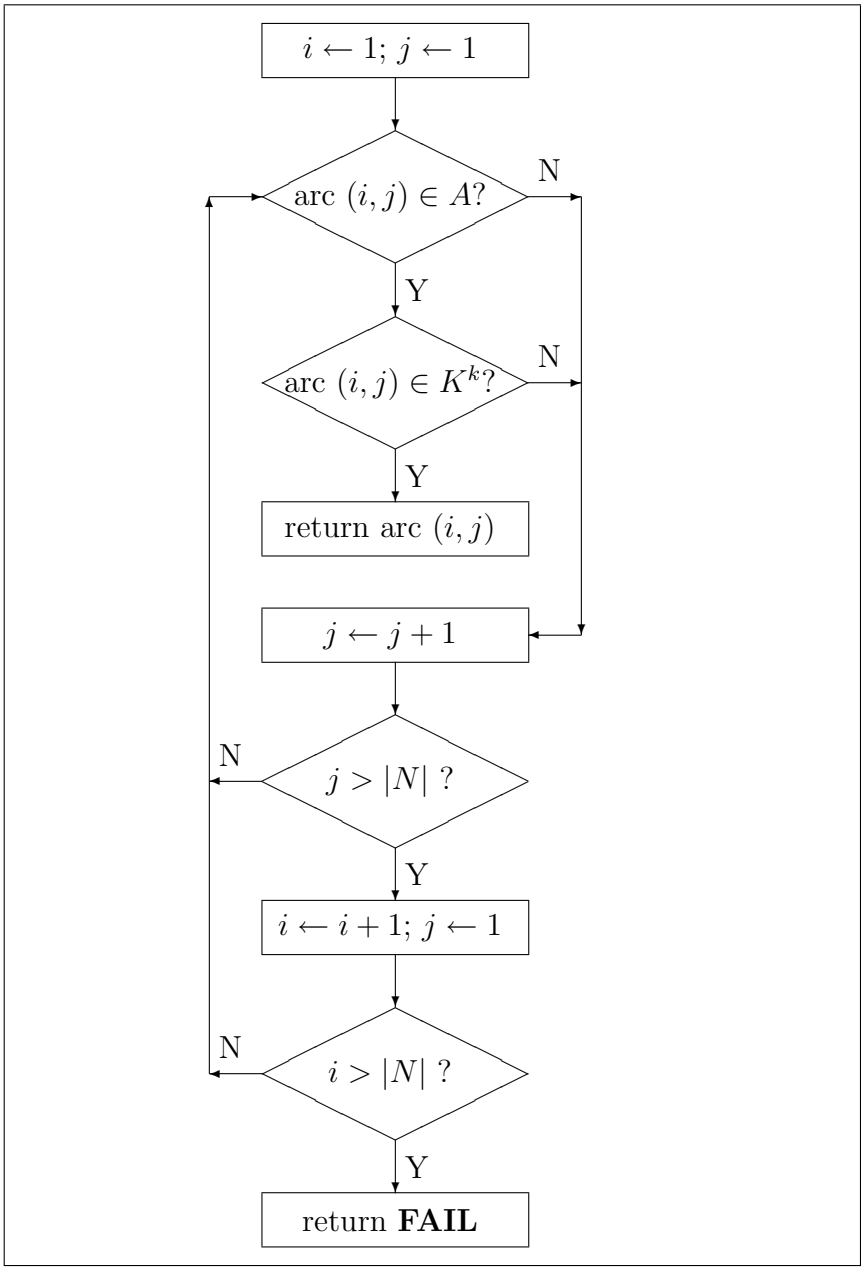

Fig. 3. "Naive" Branching Strategy

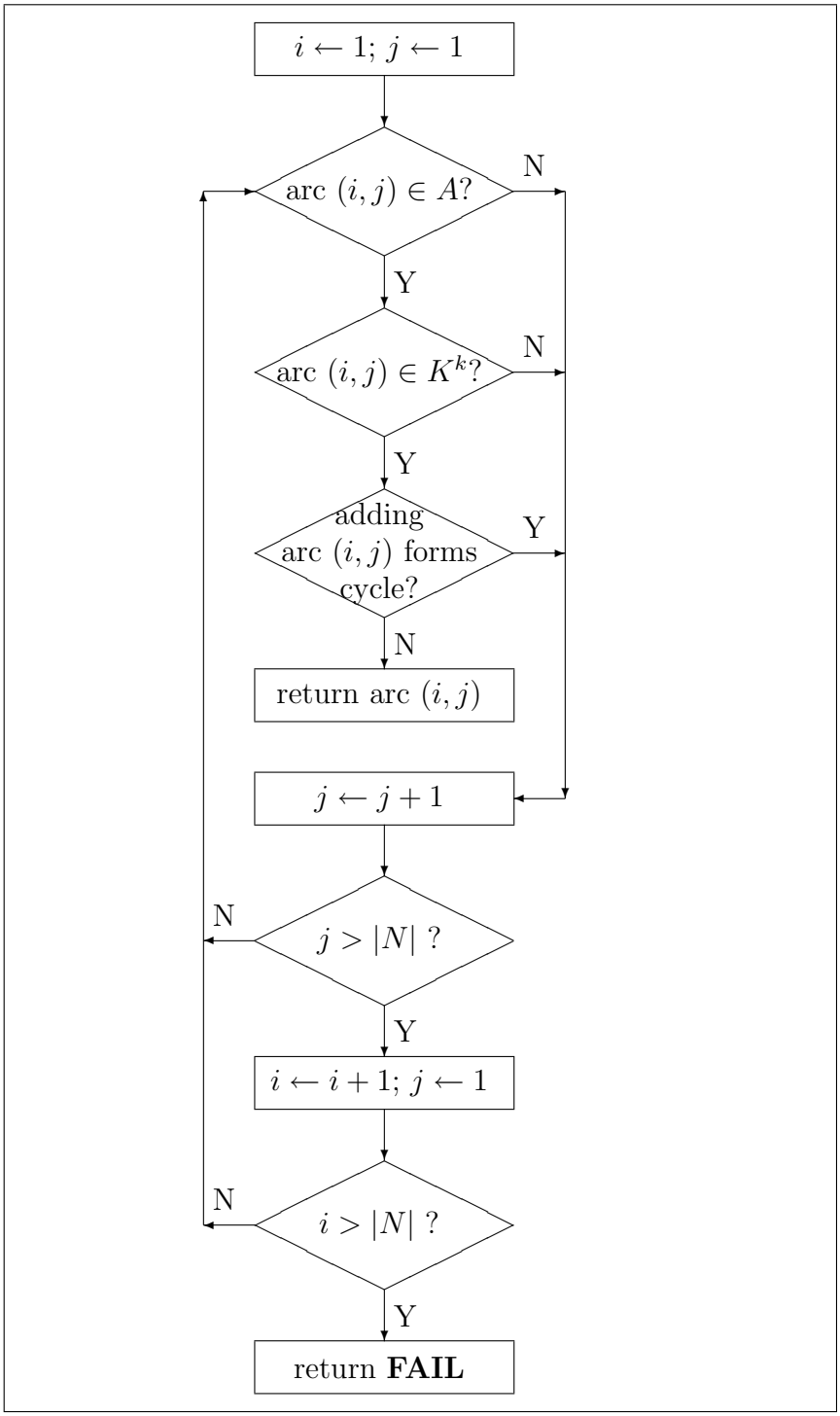

Fig. 4. Improved Branching Strategy

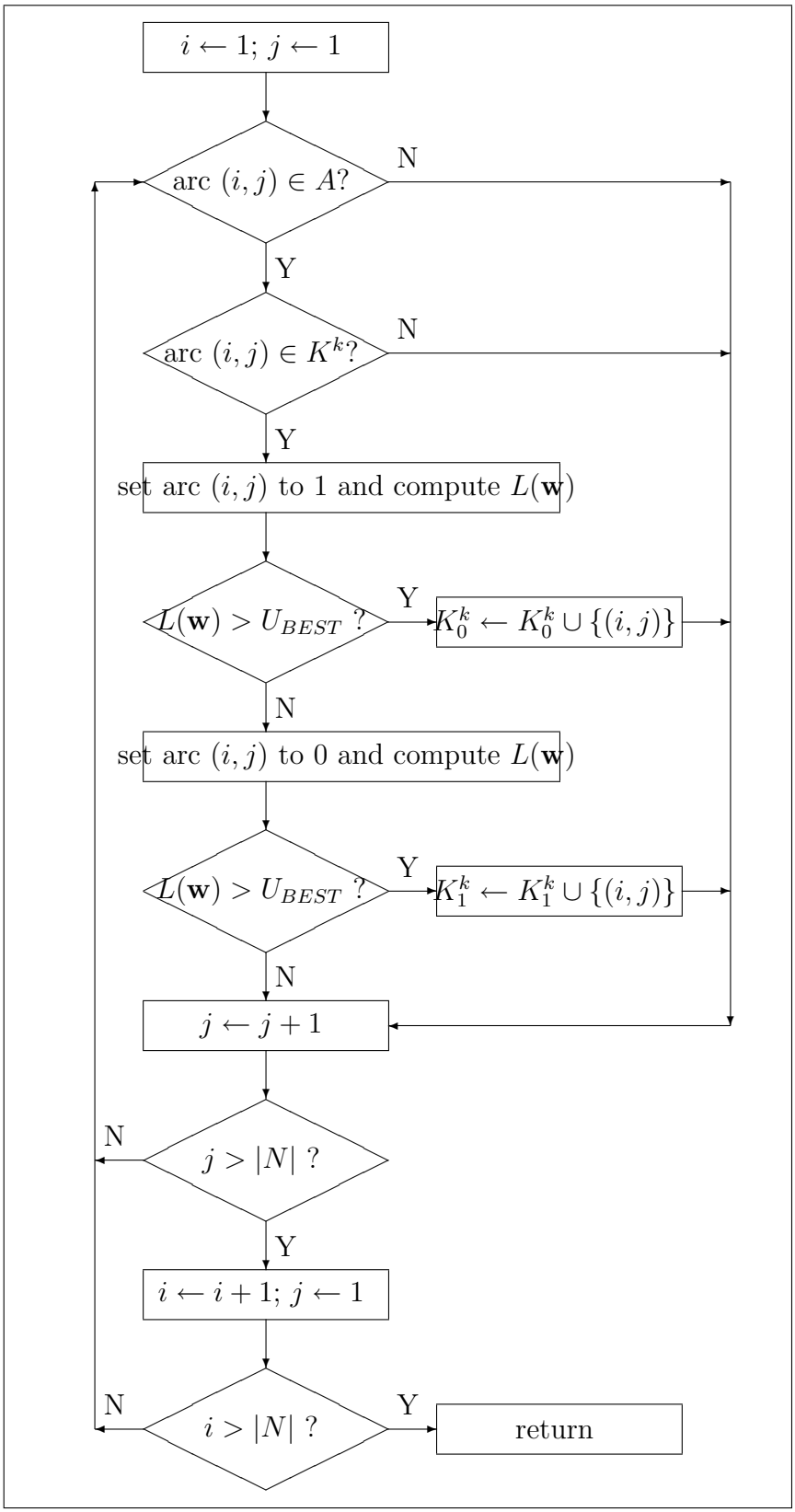

Fig. 5. Reduction Test Based on Lagrangean Relaxation