To appear in European Journal of Operational Research

Algorithms for a Multi-level Network Optimization Problem

F. R. B. Cruz[a,*], J. MacGregor Smith[b], G. R. Mateus[c]

[a]*Departamento de Estatística,*
[c]*Departamento de Ciência da Computação,*
Universidade Federal de Minas Gerais,
31270-901 - Belo Horizonte - MG, Brazil
E-mail: {fcruz@est,mateus@dcc}.ufmg.br

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

First Revision
June, 1998

Abstract

In this paper we work on a multi-level network optimization problem that integrates into the same model important aspects of (i) discrete facility location, (ii) topological network design, and (iii) network dimensioning. Potential applications for the model are discussed, stressing its growing importance. The multi-level network optimization problem treated is defined and a mathematical programming formulation is presented. We make use of a branch-and-bound algorithm based on Lagrangean relaxation lower bounds to introduce some new powerful auxiliary algorithms to exactly solve the problem. We conduct a set of computational experiments that indicate the quality of the proposed approach.

Keywords: Branch-and-bound, network programming, location, topological network design, network flows.

*To whom all correspondences should be addressed at: Caixa Postal 702, Departamento de Estatística - UFMG, 30123-970 - Belo Horizonte - MG, Brazil.

1 Introduction

The multi-level network optimization (MLNO) problem treated here was introduced recently [16], and integrates into the same model discrete location, topological network design, and dimensioning aspects. A multi-level network is illustrated in Figure 1. The MLNO problem is defined on a multi-weighted digraph $\mathcal{D} = (N, A)$ where N is the set of nodes and A is the set of arcs. There are candidate supply nodes which are grouped into m sets according to their ability to provide a specific flow type, where m is the number of levels. There are demand nodes also grouped into m sets. Similarly, all nodes in a specific set require a certain amount of the same flow type. Transshipment or *Steiner* nodes may or may not be present. The objective is to determine an optimum combination of supply nodes and arcs to provide the required flow type to all demand nodes respecting certain rules of flow conservation and transformation. The costs involved are a fixed setup cost associated with supply nodes and arcs chosen and a variable cost dependent on the amount of flow within each arc.

Figure 1 goes around here

A better comprehension about multi-level networks is relevant in theoretical terms because they can be viewed as a generalization of several important network optimization problems such as topological network design problems, fixed-charge problems, or uncapacitated location problems. A bibliography about topological network design problems in general can be found in [44]. Many network optimization problems within telecommunication applications have been studied including topological network design problems [28, 7], topological network design and dimensioning problems [38], and routing problems [5]. The fixed-charge network flow problem [45], which is a special case that represents an important class of mixed-integer programming problems, was studied in [36] and [17]. The Steiner problems in graphs [39] is probably the most studied subproblem. Although it is a classical model, recent new results are being discovered for the problem [33]. The uncapacitated location problem [22] is another relevant subproblem. The solution of this problem has many implications in the real world and recent advances continue to be made [24].

The study of multi-level models is also important in practice. Potential applications for the MLNO problem include the design of the electrical power systems interconnecting the power-stations and load centers. The MLNO problem is able to represent some of the most important technical and safety issues involved such as (i) the use of extra-high voltage lines (typically from 275 kV to 1,000 kV) for efficient electric energy transmission, (ii) the use of intermediate (11 kV) and low (220 V) voltage levels in densely occupied zones, (iii) accommodation of all consumer classes namely domestic, commercial, and industrial, and their different voltage level requirements, and (iv) problems concerning minimum safety norms for location of high voltage sub-stations and consumer sub-stations.

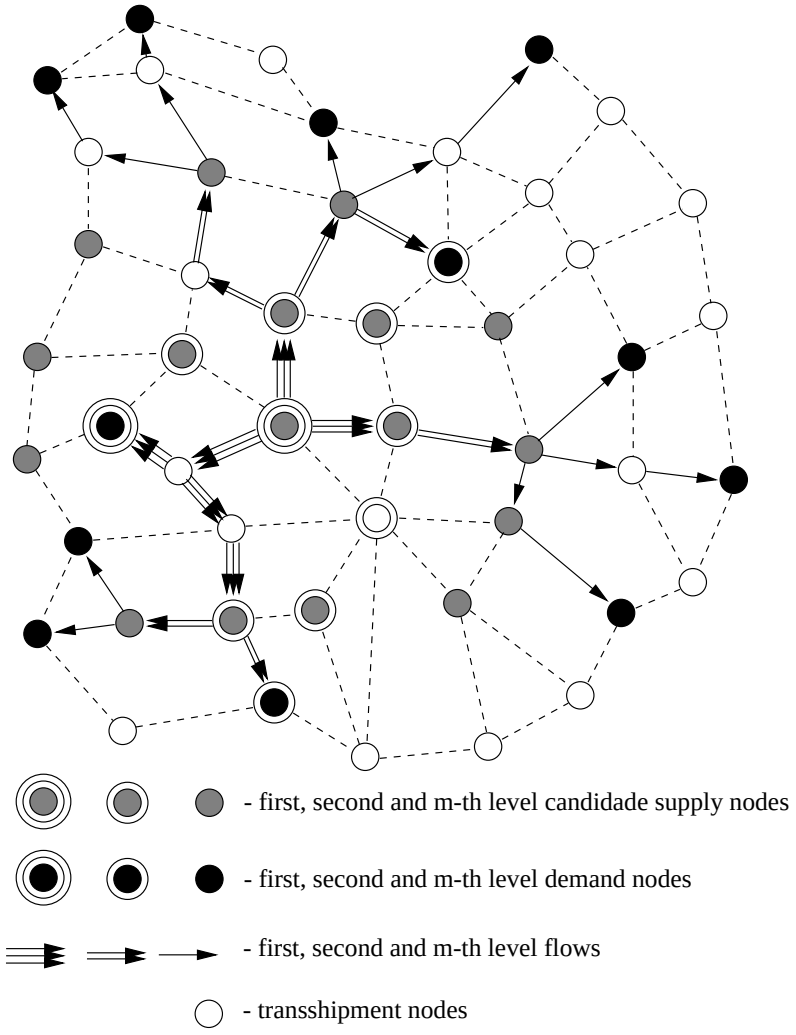

Figure 1: A Multi-level Network

Telecommunication networks are another possible application for MLNO problems. With the introduction of digital technology, new services appeared yielding mixed networks demanding different flows, *e.g.* voice simultaneously with non-voice services such as transmission of digital images and sounds in multimedia applications, high-speed data, *etc.* Additionally, different transmission mediums with different transmission rates have been coexisting for a long time, *e.g.* radio-links, optical fibers, and copper cables. The multi-level model could deal with these issues properly.

Little research has been done on the MLNO problem. Multi-level networks have appeared in some recent works but they do not consider the integration of location, design, and dimensioning aspects in the same model [19, 7, 6] nor they provide the mathematical formulation and bounds for the general problem [18, 42].

As a final remark, generally speaking, MLNO problems have been traditionally solved by methodologies that adopt a strategy of solving partial problems [43, 27, 9]. The overall problem is separated into smaller subsystems in such a way that it does not cause appreciable errors. The model we are working with here is a departure from such an approach.

The paper is outlined as follows. In Section 2, we present a mathematical programming formulation for the MLNO problem. Section 3 is devoted to all algorithms we have developed, which are a branching variable choice algorithm and a Lagrangean relaxation based reduction algorithm. Section 4 is intended to present all computational results. In this section, we try to demonstrate the usefulness of our algorithms using randomly generated problems. Section 5 presents conclusions, final remarks, discussion of open questions, and some possible extensions.

2 Problem Formulation

In the definition of the MLNO problem, we have assumed that (i) the arc cost parameters include a non-negative fixed cost of using the arc and a non-negative cost per unit of flow, (ii) there is a cost of transforming flows from one level to another, representing the costs of the hardware that must be present to interconnect the different level networks, (iii) the supply capacity of the first level candidate supply nodes equals the sum of all demands in all levels, and (iv) the candidate supply nodes of all other levels are only "transformation" nodes, receiving flows from one level and converting them to the adjacent in a 1:1 ratio.

Regarding the last assumption, we remark that the ratio 1:1 keeps the model simpler and does not make it less useful. It would be possible to model whatever ratio by convenient adjustments on the flow scales. The electrical engineers are used to do so in studying electric power systems referring all voltages to one side of the transformers.

The following notation is used in the formulation of MLNO problem:

m - number of levels;

R^l - set of l-th level candidate supply nodes;

D^l - set of l-th level demand nodes;

d_i - demand of the l-th level demand node $i \in D^l$;

T^l - set of l-th level transshipment nodes defined as follows: $T^l = N \setminus (R^l \cup D^l \cup R^{l+1})$ for $l = 1, 2, \ldots, (m-1)$, and $T^m = N \setminus (R^m \cup D^m)$;

c_{ij}^l - non-negative per unit cost for l-th level flow on arc $(i,j) \in A$;

x_{ij}^l - l-th level flow through arc $(i,j) \in A$;

f_{ij}^l - non-negative fixed cost for using arc $(i,j) \in A$ to support l-th level flow;

y_{ij}^l - boolean variable which assumes the value 1 or 0 depending on whether or not the arc (i,j) is being used to support l-th level flow;

f_i - non-negative allocation cost for the l-th level candidate supply node $i \in R^l$;

z_i - boolean variable which is set to 1 or 0 depending on whether or not the node $i \in R^l$ is being selected to provide l-th level flow;

M^l - capacity on all arcs in the l-th level, but relaxed in this paper and considered a *big* enough number, *i.e.* $M^l = \sum_{L=l}^{m} \sum_{i \in D^L} d_i$;

s^l - capacity on all l-th level candidate supply nodes, but also relaxed in this paper, *i.e.* $s^l = M^l$;

$\delta^+(i)$ - set $\{j | (i,j) \in A\}$;

$\delta^-(i)$ - set $\{j | (j,i) \in A\}$.

The MLNO problem may be described by the following mathematical programming formulation [16]:

(M)**:**

$$\min \sum_{l=1}^{m} \left[\sum_{(i,j) \in A} \left(c_{ij}^l x_{ij}^l + f_{ij}^l y_{ij}^l \right) + \sum_{i \in R^l} f_i z_i \right], \tag{1}$$

s.t.:

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l = - \left(\sum_{j \in \delta^+(i)} x_{ij}^{l-1} - \sum_{j \in \delta^-(i)} x_{ji}^{l-1} \right), \quad \forall \quad \begin{array}{l} i \in R^l, \\ l=2,3,\ldots,m, \end{array} \tag{2}$$

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l = 0, \quad \forall \quad \begin{array}{l} i \in T^l, \\ l=1,2,\ldots,m, \end{array} \tag{3}$$

$$\sum_{j\in\delta^+(i)} x^l_{ij} - \sum_{j\in\delta^-(i)} x^l_{ji} = -d_i, \quad \forall \quad \begin{array}{l} i \in D^l, \\ l=1,2,\ldots,m, \end{array} \tag{4}$$

$$\sum_{j\in\delta^+(i)} x^l_{ij} - \sum_{j\in\delta^-(i)} x^l_{ji} \leq s^l z_i, \quad \forall \quad \begin{array}{l} i \in R^l, \\ l=1,2,\ldots,m, \end{array} \tag{5}$$

$$x^l_{ij} \leq M^l y^l_{ij}, \quad \forall \quad \begin{array}{l} (i,j) \in A, \\ l=1,2,\ldots,m, \end{array} \tag{6}$$

$$x^l_{ij} \geq 0, \quad \forall \quad \begin{array}{l} (i,j) \in A, \\ l=1,2,\ldots,m, \end{array} \tag{7}$$

$$y^l_{ij} \in \{0,1\}, \quad \forall \quad \begin{array}{l} (i,j) \in A, \\ l=1,2,\ldots,m, \end{array} \tag{8}$$

$$z_i \in \{0,1\}, \quad \forall \quad \begin{array}{l} i \in R^l, \\ l=1,2,\ldots,m. \end{array} \tag{9}$$

The objective function (1) minimizes (i) the total variable flow cost for all levels, (ii) the total fixed cost associated with the use of the arcs, and (iii) the total fixed cost resulting from the use of the candidate supply nodes. Constraints (2) ensure the network flow conservation between adjacent levels at each candidate supply node, constraints (3), and (4) are the usual network flow conservation equalities at each *Steiner* or transshipment node and at each demand node. Constraints (5) ensure there is no flow transformation in a candidate supply node if it is not selected, and constraints (6) express the fact that the flow through an arc must be zero if this arc is not included in the design.

In this paper, we shall define some auxiliary sets that will be helpful in describing the proposed algorithms. So let us define $J_0 \subseteq N$, the set of nodes that have been fixed as not-present in an optimal solution by some algorithm, $J_1 \subseteq N$, the set of nodes that have been fixed as present in an optimal solution, and $J = N \setminus (J_0 \cup J_1)$, the set of free or not fixed nodes. Similarly, let us define $K^l_0 \subseteq A$, the set of arcs fixed as not-present in the l-th level, $K^l_1 \subseteq A$, the set of arcs fixed as present, and $K^l = A \setminus (K^l_0 \cup K^l_1)$, the set of free or undefined arcs. The formulation (M), re-written and incorporating these new sets, is able to represent either the problem after the application of the reduction test as well as all subproblems created by the branch-and-bound algorithm, as presented in the next section.

3 Algorithm Descriptions

The multi-level network optimization problem is $\mathcal{NP}$-hard since it generalizes other $\mathcal{NP}$-hard optimization problems such as the Steiner problem in graphs [26] or the uncapacitated location problem [22]. If the model (M) had only the first level, one supply node, null costs c^l_{ij} and f_i, and more than two demand nodes, it would be simply representing a Steiner problem in graphs. Thus, solving the MLNO problem means solving an embedded Steiner ($\mathcal{NP}$-hard) problem.

Considering its $\mathcal{NP}$-hardness, one possibility to solve the MLNO problem is to try to find heuristic algorithms that give *good* solutions within an acceptable amount of time. An example

of a widely applied technique is *local search* in which preselected operations are used in order to improve an initial solution continuing until no further improvements can be made (*e.g.* simulated annealing, [1], and tabu search, [31, 32]). Although rarely, it is possible to predict how well heuristic algorithms will perform in practice by formally analyzing them beforehand. A number of results in this area has been obtained for similar problems [3, 37, 11, 34, 2]. Such results are viewed as proving *performance guarantees* for heuristic algorithms, which is a vast research field.

We shall concentrate in another alternative. After acknowledging the apparent inevitability of exponential time complexity to prove optimality, we will seek to obtain as much improvement over straightforward exhaustive search as possible. Among the most widely used approaches to reduce the searching effort are those based on *branch-and-bound* techniques.

The template of the non-recursive version used here is depicted in Figure 2. In that description, U_{BEST} is the global upper bound and Γ is a list of unexplored problems $(M)^i$, each of which is of the form $Z_M^i = \min\{\mathbf{cx}\ \mathbf{s.t.:}\ \mathbf{x} \in S^i\}$, such that $S^i \subseteq S$ and $S^i \cup S^{i+1} = S$, where S is the set of feasible solutions. Associated with each problem in Γ are a lower bound $L^i \leq Z_M^i$ and an upper bound $U^i \geq Z_M^i$. For memory economy purposes, the search rule applied was *last-in-first-out* which yields a *depth-first* search strategy.

Figure 2 goes around here

```
algorithm Branch-and-Bound
    U_BEST ← +∞
    Γ ← {(M)^0}
    while Γ ≠ ∅ do
            /* search rule */
        select and delete a problem (M)^i from Γ
            /* bound rule */
        Compute_Lower_and_Upper_Bounds(L^i,U^i)
        update U_BEST
            /* branch rule */
        Reduce[(M)^i]
        if L^i < U_BEST and (M)^i is not a leaf then
            Γ ← Γ ∪ {(M)^(2i+1)} ∪ {(M)^(2i+2)}
        end if
    end while
end algorithm
```

Figure 2: Branch-and-Bound Algorithm

3.1 Lower Bounds

A well-known technique to derive lower bounds is the Lagrangean relaxation, which is usually coupled with subgradient optimization procedures [23]. The Lagrangean has been successfully applied in many similar combinatorial optimization problems, such as location problems [10, 25,

41], design of distribution systems [30], traveling salesman problems [35], design of computer networks [27, 28], and to the MLNO problem itself [16], with promising results.

There are many ways to derive a Lagrangean relaxation for model (M). It was proposed to drop constraints (5) , by using the Lagrangean multipliers $v_i \geq 0$, and constraints (6), by using Lagrangean multipliers $w_{ij}^l \geq 0$. Then, the Lagrangean function is:

$$\begin{aligned}\mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{z};\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m) &= \sum_{l=1}^{m}\left[\sum_{(i,j)\in A}\left(c_{ij}^l x_{ij}^l + f_{ij}^l y_{ij}^l\right) + \sum_{i\in R^l} f_i z_i\right] + \\ &\sum_{l=1}^{m}\sum_{i\in R^l\cap J} v_i\left(\sum_{j\in\delta^+(i)} x_{ij}^l - \sum_{j\in\delta^-(i)} x_{ji}^l - s^l z_i,\right) + \\ &\sum_{l=1}^{m}\sum_{(i,j)\in K^l} w_{ij}^l\left(x_{ij}^l - M^l y_{ij}\right), \end{aligned} \tag{10}$$

which results in the following Lagrangean relaxation:

$(LR_{\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m})$**:**

$$L(\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m) = \min_{\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m\geq\mathbf{0}} \mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{z};\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m), \tag{11}$$

s.t.:

(2)–(4), (7)–(9).

Supposing that $L(\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m) = \mathcal{L}(\mathbf{x}^*,\mathbf{y}^*,\mathbf{z}^*;\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m)$, the subgradient vector $\boldsymbol{\gamma}$ of the function L at point $(\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m)$ is:

$$\boldsymbol{\gamma} = \left[\left(\sum_{j\in\delta^+(i)} {x_{ij}^l}^* - \sum_{j\in\delta^-(i)} {x_{ji}^l}^* - s^l {z_i}^*,\right)_{\substack{i\in R^l\cap J,\\ l=1,2,\ldots,m}}, \left({x_{ij}^l}^* - M^l {y_{ij}^l}^*\right)_{\substack{(i,j)\in K^l,\\ l=1,2,\ldots,m}}\right]. \tag{12}$$

Once feasible values for the Lagrangean multipliers $\mathbf{v}$, $\mathbf{w}^1$, $\mathbf{w}^2$,..., and $\mathbf{w}^m$ are given, the computation of the function $L(\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m)$ is reduced to solve easy subproblems (in such context, *easy* is used in reference to polynomially solvable problems):

$$\begin{aligned} L(\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m) &= L_1(\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m) + \\ &L_2(\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m) + \\ &L_3(\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m), \end{aligned} \tag{13}$$

where $L_1(\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m)$, $L_2(\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m)$, and $L_3(\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m)$ are optimal solutions of the subproblems shown below.

Subproblem (L_1)

The subproblem (L_1) is:

(L_1)**:**

$$L_1(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^m) = \min \sum_{l=1}^{m} \sum_{(i,j)\in A} C_{ij}^l x_{ij}^l, \tag{14}$$

s.t.:

$$\sum_{j\in\delta^+(i)} x_{ij}^l - \sum_{j\in\delta^-(i)} x_{ji}^l = -\left(\sum_{j\in\delta^+(i)} x_{ij}^{l-1} + \sum_{j\in\delta^-(i)} x_{ji}^{l-1} \right), \quad \forall \quad \begin{array}{l} i \in R^l, \\ l=2,3,\ldots,m, \end{array} \tag{15}$$

$$\sum_{j\in\delta^+(i)} x_{ij}^l - \sum_{j\in\delta^-(i)} x_{ji}^l = 0, \quad \forall \quad \begin{array}{l} i \in T^l, \\ l=1,2,\ldots,m, \end{array} \tag{16}$$

$$\sum_{j\in\delta^+(i)} x_{ij}^l - \sum_{j\in\delta^-(i)} x_{ji}^l = -d_i, \quad \forall \quad \begin{array}{l} i \in D^l, \\ l=1,2,\ldots,m, \end{array} \tag{17}$$

$$\sum_{j\in\delta^+(i)} x_{ij}^l - \sum_{j\in\delta^-(i)} x_{ji}^l \leq s^l, \quad \forall \quad \begin{array}{l} i \in R^l \cap J_1, \\ l=1,2,\ldots,m, \end{array} \tag{18}$$

$$\sum_{j\in\delta^+(i)} x_{ij}^l - \sum_{j\in\delta^-(i)} x_{ji}^l = 0, \quad \forall \quad \begin{array}{l} i \in R^l \cap J_0, \\ l=1,2,\ldots,m, \end{array} \tag{19}$$

$$x_{ij}^l \leq M^l, \quad \forall \quad \begin{array}{l} (i,j) \in K_1^l, \\ l=1,2,\ldots,m, \end{array} \tag{20}$$

$$x_{ij}^l \leq 0, \quad \forall \quad \begin{array}{l} (i,j) \in K_0^l, \\ l=1,2,\ldots,m, \end{array} \tag{21}$$

$$x_{ij}^l \geq 0, \quad \forall \quad \begin{array}{l} (i,j) \in A, \\ l=1,2,\ldots,m, \end{array} \tag{22}$$

where

$$C_{ij}^l = \begin{cases} +\infty & (i,j) \in K_0^l, \\ c_{ij}^l & (i,j) \in K_1^l, \quad j \notin R^l \cap J, \quad i \notin R^l \cap J, \\ c_{ij}^l + v_i & (i,j) \in K_1^l, \quad j \notin R^l \cap J, \quad i \in R^l \cap J, \\ c_{ij}^l - v_j, & (i,j) \in K_1^l, \quad j \in R^l \cap J, \quad i \notin R^l \cap J, \\ c_{ij}^l + v_i - v_j, & (i,j) \in K_1^l, \quad j \in R^l \cap J, \quad i \in R^l \cap J, \\ c_{ij}^l + w_{ij}^l & (i,j) \in K^l, \quad j \notin R^l \cap J, \quad i \notin R^l \cap J, \\ c_{ij}^l + w_{ij}^l + v_i & (i,j) \in K^l, \quad j \notin R^l \cap J, \quad i \in R^l \cap J, \\ c_{ij}^l + w_{ij}^l - v_j, & (i,j) \in K^l, \quad j \in R^l \cap J, \quad i \notin R^l \cap J, \\ c_{ij}^l + w_{ij}^l + v_i - v_j, & (i,j) \in K^l, \quad j \in R^l \cap J, \quad i \in R^l \cap J. \end{cases} \tag{23}$$

The optimum of (L_1) is reachable using a shortest path algorithm. The problem can be solved level by level. The optimum for the first level is to connect nodes in D^1 to nodes in R^1 using the shortest paths and avoiding arcs in set K_0^1. For the second level, the optimum is to connect nodes in D^2 also to nodes in R^1 via shortest paths but necessarily using some node in set R^2, avoiding arcs in sets K_0^1 and K_0^2, and so on for all remaining levels. The complete algorithm is seen in Figure 3.

Figure 3 goes around here

```
algorithm
        /* σ_i^l is the minimum per unit cost to bring */
        /* l-th level flow from set R^1 to node i ∈ N; */
        /*   function SH(i, j, l) returns the shortest  */
        /*   path length from i to j using costs C_ij^l; */
    L1 ← 0
    for ∀j ∈ R^1 do
        if j ∈ J then
            σ_j^0 ← 0
        else
            σ_j^0 ← +∞
        end if
    end for
    for l ← 1 to m do
        for ∀j ∈ D^l do
            σ_j^l ← min_{i∈R^l} [σ_i^{l-1} + SH(i, j, l)]
            L1 ← L1 + σ_j^l * d_j
        end for
        if l ≠ m do
            for ∀j ∈ R^{l+1} do
                if j ∈ J then
                    σ_j^l ← min_{i∈R^l} [σ_i^{l-1} + SH(i, j, l)]
                else
                    σ_j^l ← +∞
                end if
            end for
        end if
    end for
end algorithm
```

Figure 3: Algorithm for Problem (L_1)

The shortest *simple* path algorithm for arbitrary costs has a worst case time complexity $O(|N|\ |A|)$ if there are no negative cost circuits [12]. Thus, from Theorem 1 [16], the algorithm for solving problem (L_1) presented in Figure 3 efficiently implemented has worst case time complexity $O(m\ |N|\ |A|)$ which translates to $O(m\ |N|^3)$ in dense networks.

Theorem 1 *The problem* (L_1) *on graph* $G = (N, A)$ *with weights as defined in Equation (23) does not have negative cost circuits.*

Proof (by construction): *Let* C^l *be an arbitrary circuit in the* l*-th level,* $A(C^l) \subseteq A$ *be the set of arcs in that circuit and* $N(C^l) \subseteq N$ *be the set of nodes in the circuit. From Equation (23), the per unit cost associated with circuit* C^l *must be non-negative:*

$$\begin{aligned}\sum_{(i,j)\in A(C^l)} C_{ij}^l &= \sum_{(i,j)\in A(C^l)} c_{ij}^l + \sum_{(i,j)\in A(C^l)\cap K^l} w_{ij}^l + \sum_{i\in N(C^l)\cap R^l\cap J} v_i - \sum_{j\in N(C^l)\cap R^l\cap J} v_j \\ &= \sum_{(i,j)\in A(C^l)} c_{ij}^l + \sum_{(i,j)\in A(C^l)\cap K^l} w_{ij}^l\end{aligned}$$

$$\geq \quad 0.$$

■

Subproblem (L_2)

The subproblem (L_2) is a subset selection problem:

(L_2)**:**

$$L_2(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^m) = \min \sum_{l=1}^{m} \sum_{(i,j)\in A} \left(f_{ij}^l - w_{ij}^l M^l\right) y_{ij}^l, \tag{24}$$

s.t.:

$$y_{ij}^l \in \{0,1\}, \quad \forall \quad \underset{l=1,2,\ldots,m,}{(i,j) \in K^l}, \tag{25}$$

$$y_{ij}^l = 1, \quad \forall \quad \underset{l=1,2,\ldots,m,}{(i,j) \in K_1^l}, \tag{26}$$

$$y_{ij}^l = 0, \quad \forall \quad \underset{l=1,2,\ldots,m,}{(i,j) \in K_0^l}, \tag{27}$$

which can be solved by an algorithm with time complexity $O(m|A|)$, or $O(m|N|^2)$ for dense networks.

Subproblem (L_3)

Similarly, the problem (L_3) is also a subset selection problem:

(L_3)**:**

$$L_3(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^m) = \min \sum_{l=1}^{m} \sum_{i\in R^l} \left(f_i - v_i s^l\right) z_i, \tag{28}$$

s.t.:

$$z_i \in \{0,1\}, \quad \forall \quad \underset{l=1,2,\ldots,m,}{i \in R^l \cap J}, \tag{29}$$

$$z_i = 1, \quad \forall \quad \underset{l=1,2,\ldots,m,}{i \in R^l \cap J_1}, \tag{30}$$

$$z_i = 0, \quad \forall \quad \underset{l=1,2,\ldots,m,}{i \in R^l \cap J_0}, \tag{31}$$

solvable by an algorithm with time complexity $O(|N|)$.

3.2 Upper Bounds

There are many possibilities of computing an upper bound for model (M). The method proposed here takes advantage of the problem (L_1) optimal solution. Actually, this solution may be turned into a feasible solution for model (M) if feasibility of the previously dropped constraints (5) and (6) is ensured. Using the same arcs as in the optimal solution of problem (L_1), we have just to add to the flow costs c_{ij}^l, the fixed costs of the used arcs f_{ij}^l, and the fixed costs of the used supply nodes f_i. Clearly, the proposed heuristic is polynomial.

3.3 Branching Strategy

In order to generate the child problems $(M)^{2i+1}$ and $(M)^{2i+2}$, the simplest branching strategy is to choose the first free decision variable, Figure 4. It is easily seen that the worst case time complexity of such procedure is $O(m(|N| + |A|))$.

Figure 4 goes around here

```
procedure Chosen_Arc_First_Free
    for l = 1 to m do
        /* try to choose a node */
        for ∀i ∈ R^l do
            if i ∈ J then
                return i
            end if
        end for
        /* otherwise try to choose an arc */
        for ∀(i, j) ∈ A do
            if (i, j) ∈ K^l then
                return (i, j), l
            end if
        end for
    end for
    return FAIL
end procedure
```

Figure 4: "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 MLNO problems have an important property that can be very helpful. The property, stated by the Theorem 2, is an extension of a similar result obtained for the fixed-charge network flow problem [17].

Theorem 2 *If the MLNO problem has an optimum solution then there is an optimum positive flow arc set in such way that at most one arc of each level enters into each node.*

Proof (by contradiction): *Let us suppose that Theorem 2 is not satisfied by any optimal solution and that node u has two entering arcs in the l-th level say (s, u) and (t, u). There must be a set of arcs in the optimal solution forming a directed path without cycles from some l-th level candidate supply node to node u passing through node s,* i.e. *using arc (s, u), called P_s. There must be also a directed path without cycles using arc (t, u), called P_t. Without loss of generality, let us suppose that*

$$\sum_{(i,j)\in P_s} c_{ij}^l \leq \sum_{(i,j)\in P_t} c_{ij}^l.$$

Thus, disabling arc (t,u) and transferring its flow, x_{tu}, from path P_t to path P_s there will be at least the following reduction in the objective function:

$$\left(\sum_{(i,j)\in P_t} c_{ij}^l - \sum_{(i,j)\in P_s} c_{ij}^l \right) x_{ij}^l + f_{tu}^l \geq 0.$$

The resulting solution is at least as good as the original and satisfies Theorem 2, which contradicts our initial assumption. ■

Thus, a possibly much better algorithm is to choose the first free decision variable that does not violate Theorem 2, Figure 5.

Figure 5 goes around here

```
procedure Chosen_Arc_No_Cycling
        /* try to choose a first level candidate supply node */
    for ∀i ∈ R^1 do
        if i ∈ J then
            return i
        end if
    end for
        /* otherwise go throughout all levels *
    for l = 1 to m do
            /* try to choose a (l + 1)-th level candidate supply node */
        for ∀i ∈ R^{l+1} do
            if i ∈ J then
                return i
            end if
        end for
            /* otherwise try to choose an arc in the l-th level */
                /* compute set of reached nodes T */
        T ← ∅
        for ∀i ∈ N do
            if i ∈ R^l ∩ J_1 then
                T ← T ∪ {i}
            end if
        end for
        for ∀(i, j) ∈ A do
            if (i, j) ∈ K_1^l then
                T ← T ∪ {i} ∪ {j}
            end if
        end for
                /* search new eligible free arc */
        if there is i ∈ D^l ∪ (R^{l+1} ∩ J_1) such that i ∉ T then
            for ∀(i, j) ∈ A do
                if (i, j) ∈ K^l and i ∈ T and j ∉ T then
                    return (i, j), l
                end if
            end for
        end if
    end for
    return FAIL
end procedure
```

Figure 5: Improved Branching Strategy

The choice of a candidate supply node is decided in at most $O(|N|)$ operations. The computation of set T may be done $O(|N|+|A|)$. The choice of an arc is decided in at most $O(|N|+|A|)$

operations noting that the statement "**if** *there is* $i \in D^l \cup (R^{l+1} \cap J_1)$ *such that* $i \notin T$ **then**", Figure 5, may be done $O(|N|)$. So, an $O(m(|N| + |A|))$ overall worst case time complexity results, surprisingly the same as in the *Chosen_Arc_First_Free* procedure. Thus, using the *Chosen_Arc_No_Cycling* procedure is at most as bad as using the former procedure.

3.4 Reduction Procedure

The new lower bound that would result from forcing the nodes and arcs *in* or *out of* the solution can be easily estimated from the Lagrangean relaxation. If the lower bound resulting from imposing some condition of the Lagrangean relaxation is above the best upper bound, then the condition in consideration cannot be satisfied in the optimum. This idea is an extension of the reduction procedures developed in [17] to solve the fixed-charge networks flow problem, with very good results in practice. In [14], some terms of the corresponding Lagrangean function have been used to estimate the increment in the lower bound under the imposed condition. We propose a reduction procedure that uses lower bounds computed by means of a complete resolution of the Lagrangean dual problem, $(LR_{\mathbf{v},\mathbf{w}^1,\mathbf{w}^2,\ldots,\mathbf{w}^m})$, but without subgradient optimizations. The algorithm is presented in Figure 6.

Figure 6 goes around here

In the worst case, the loop "**repeat** *statements* **until** *condition*", Figure 6, stops after $O(|N| + m|A|)$ iterations considering the worst case in which only one reduction occurs per iteration. The statement "**for** $\forall i \in R^l \cap J$ **do** *statements* **end for**" is $O(|N|m|N||A|)$ since it involves $O(|N|)$ lower bound computations which are $O(m|N||A|)$. Similarly, the statement "**for** $\forall (i,j) \in K^l$ **do** *statements* **end for**" is $O(|A|m|N||A|)$. So, the resulting worst case time complexity of the reduction algorithm is $O\left(m^2|N||A|(|N| + m|A|)(|N| + |A|)\right)$ which is polynomial.

4 Computational Experience

All algorithms developed here were coded in *C* and are available from the authors upon request. All tests were performed with the use of a workstation *Sun* Ultra 1 Model 140, RAM 128 MB, running the *Sun* operating system SunOS, Rel. 5.5.1. All CPU times reported are in seconds, excluding all I/O operations and considering that the processor was dedicated to solve the instance.

The test problems were randomly generated by the following algorithm. Using the uniform probability distribution, $|N|$ nodes were allocated in a 100×100 square and $|N| - 1$ edges were defined to ensure a connected network. Additional edges up to the number $|A|/2$ were generated. Finally, the undirected graph resulting was converted into a directed graph replacing each edge by two opposite side arcs. *Basic* weights Ω_{ij} were defined as the Euclidean distance between the extremities i and j.

```
procedure Reduce(M)
        /* initialize set of nodes and arcs recently fixed */
    F_J ← ∅; F_{K^l} ← ∅, l = 1, 2, ..., m
        /* proceed with reduction */
    repeat
        for l = 1 to m do
                /* try to fix free supply nodes */
            for ∀i ∈ R^l ∩ J do
                J ← J \ {i}; J_1 ← J_1 ∪ {i}
                if L(v, w^1, w^2, ..., w^m) > U_BEST then
                    J_1 ← J_1 \ {i}; J_0 ← J_0 ∪ {i}; F_J ← F_J ∪ {i}
                else
                    J_1 ← J_1 \ {i}; J_0 ← J_0 ∪ {i}
                    if L(v, w^1, w^2, ..., w^m) > U_BEST then
                        J_0 ← J_0 \ {i}; J_1 ← J_1 ∪ {i}; F_J ← F_J ∪ {i}
                    else
                        J_0 ← J_0 \ {i}; J ← J ∪ {i}
                    end if
                end if
            end for
                /* try to fix free arcs */
            for ∀(i, j) ∈ K^l do
                K^l ← K^l \ {(i, j)}; K_1^l ← K_1^l ∪ {(i, j)}
                if L(v, w^1, w^2, ..., w^m) > U_BEST then
                    K_1^l ← K_1^l \ {(i, j)}; K_0^l ← K_0^l ∪ {(i, j)}; F_{K^l} ← F_{K^l} ∪ {(i, j)}
                else
                    K_1^l ← K_1^l \ {(i, j)}; K_0^l ← K_0^l ∪ {(i, j)}
                    if L(v, w^1, w^2, ..., w^m) > U_BEST then
                        K_0^l ← K_0^l \ {(i, j)}; K_1^l ← K_1^l ∪ {(i, j)}; F_{K^l} ← F_{K^l} ∪ {(i, j)}
                    else
                        K_0^l ← K_0^l \ {(i, j)}; K^l ← K^l ∪ {(i, j)}
                    end if
                end if
            end for
        end for
    until no reduction has occurred
end procedure
```

Figure 6: Reduction Algorithm

This procedure is similar to that one presented by Aneja [4] that has been extensively used since then for creating random testing instances for the *Steiner* problem in graphs [46, 13, 21]. The uniform probability distribution was also used to compose the sets D^l and R^l, $l = 1, 2, \ldots, m$, with the respective cardinalities, each node being chosen at most once. All demands were considered unitary. The costs f_{ij}^l and c_{ij}^l actually used were derived from the basic weights Ω_{ij} using constant factors as presented in Table 1.

The parameters $|N|$, $|A|$, $|R^1|$, $|D^1|$, $|R^2|$, $|D^2|$, f_{ij}^l, c_{ij}^l, and f_i assumed several values, thus trying to cover different networks. The objective is to provide a simplified reference table which one could use to estimate the actual processing time for the specific network he has on hand. The second level costs were considered greater than the first level costs, which is a reasonable assumption in multi-level networks. Usually, the higher levels take advantage of concentrated

demands and are allowed to use special medium with lower costs. On the other hand, the lower levels usually must use less efficient mediums with higher costs. For a more detailed study on one-level networks involving different ratios between fixed and variable costs, see [17].

Table 1 presents the parameters used for each problem tested, the solution obtained in the first node of the search tree, the GAP $= 100\% \times (U_{\text{BEST}} - L^0)/U_{\text{BEST}}$, and the CPU time spent in the first node. The last column shows the optimal solutions obtained after the hole search. It is remarkable that only in 2 cases (problems 6 and 32, Table 1) the optimal solution was not obtained in the first node. Besides offering large gaps, mainly in those problems with large f_i, the Lagrangean relaxation based heuristics proves to be effective for solving the MLNO problem instances shown in Table 1.

Table 2 presents a comparison between the two branching strategies and the remarkable effect of the reduction algorithm working under the *Chosen_Arc_No_Cycling* procedure. The combined use of the reduction algorithm and the *Chosen_Arc_No_Cycling* procedure kept the number of explored branch-and-bound nodes surprisingly low as well as the CPU time. As an overall gain, larger instances became tractable by the workstation used.

Finally, the last column shows the CPU times obtained by means of the commercial software CPLEX [15], running in the same machine used to perform the previous tests. The results seem to indicate that our approach is competitive. Besides, the procedures developed here could be adapted and incorporated into commercial softwares as CPLEX, possibly improving their performance as well.

Table 1 goes around here

Table 2 goes around here

5 Conclusions

In this paper, we have dealt with a multi-level network optimization (MLNO) problem and stressed its importance in theoretical and practical terms. The problem was formulated and a branch-and-bound algorithm based on Lagrangean relaxation was used to introduce a new strategy for branching based on an optimum solution property and to develop a new reduction test based on Lagrangean relaxation. Although computationally inefficient with exponential worst case time complexity, the branch-and-bound algorithm may be quite useful in practice for certain sized problem instances as our computational results have demonstrated. Additionally, it was proved that the branching strategy and the reduction algorithm can accommodate larger problem instances.

Some questions remain open such as whether there would be better methods for generating the bounds the branch-and-bound algorithm uses, better branching strategies, or even more powerful

Table 1: Results on the First Node

												First Node			Branch-and-Bound
Problem	$\|N\|$	$\|A\|$	$\|R^1\|$	$\|D^1\|$	$\|R^2\|$	$\|D^2\|$	$\frac{f^1_{ij}}{\Omega_{ij}}$	$\frac{c^1_{ij}}{\Omega_{ij}}$	$\frac{f^2_{ij}}{\Omega_{ij}}$	$\frac{c^2_{ij}}{\Omega_{ij}}$	f_i	U_{BEST}	GAP(%)	CPU(s)	U_{opt}
1	4	6	1	1	1	1	1	8	2	128	0	9,732	0.10	0.02	9,732
2											8	9,732	0.10	0.02	9,732
3											1024	11,780	0.08	0.02	11,780
4		12	1	1	1	1	1	8	2	128	0	12,930	0.20	0.04	12,930
5											8	12,946	0.20	0.04	12,946
6											1024	15,007	0.37	0.05	14,978**
7	8	14	1	1	1	4	1	8	2	128	0	35,232	0.89	0.12	35,232
8											8	35,248	0.89	0.06	35,248
9											1024	37,280	0.84	0.06	37,280
10					2	4	1	8	2	128	0	26,300	0.95	0.06	26,300
11											8	26,316	0.94	0.06	26,316
12											1024	28,348	0.88	0.07	28,348
13		28	1	1	1	4	1	8	2	128	0	28,304	1.20	0.13	28,304
14											8	28,320	1.20	0.13	28,320
15											1024	30,352	1.10	0.13	30,352
16					2	4	1	8	2	128	0	27,685	1.20	0.14	27,685
17											8	27,709	1.30	0.14	27,709
18											1024	30,757	4.50	0.14	30,757
19		56	1	1	2	4	1	8	2	128	0	17,724	1.60	0.37	17,274
20											8	17,748	1.60	0.38	17,748
21											1024	20,796	6.30	0.38	20,796
22	16	30	1	1	2	4	1	8	2	128	0	38,753	0.93	0.19	38,753
23											8	38,769	0.93	0.19	38,769
24											1024	40,801	0.88	0.19	40,801
25					4	4	1	8	2	128	0	37,345	1.10	0.21	37,345
26											8	37,369	1.10	0.21	37,369
27											1024	40,417	3.60	0.22	40,417
28					8	4	1	8	2	128	0	62,241	0.33	0.24	62,241
29											8	62,257	0.33	0.24	62,257
30											1024	64,289	0.32	0.24	64,289
31		60	1	1	4	4	1	8	2	128	0	24,375	1.50	0.54	24,375
32											8	24,399	1.60	0.54	27,102**
33											1024	27,447	5.10	0.63	27,447
34					8	4	1	8	2	128	0	25,418	1.80	0.61	25,418
35											8	25,458	1.90	0.61	25,458
36											1024	30,538	11.00	1.10	30,538
37		120	1	1	2	4	1	8	2	128	0	16,609	1,20	1.70	16,609
38											8	16,633	1,30	1.70	16,633
39											1024	19,681	6,20	1.70	19,681

** $U_{\text{opt}} \neq U_{\text{BEST}}$

Table 2: Comparisons for all Algorithms

	Branch-and-Bound							
	Chosen_Arc_First_Free		*Chosen_Arc_No_Cycling*		Plus Reduction		CPLEX	
Problem	Nodes	CPU(s)	Nodes	CPU(s)	Nodes	CPU(s)	Nodes	CPU(s)
1	11	0.09	9	0.06	1	< 0.01	0	0.01
2	11	0.09	9	0.06	1	< 0.01	0	0.01
3	11	0.09	9	0.06	1	< 0.01	0	0.01
4	21	0.40	15	0.22	3	0.01	3	0.02
5	21	0.34	15	0.22	3	0.01	3	0.01
6	21	0.35	15	0.22	5	0.04	3	0.01
7	6,655	170.00	45	1.10	1	0.01	0	0.01
8	6,655	170.00	45	1.10	1	0.01	0	0.01
9	6,655	170.00	45	1.10	1	0.01	0	0.01
10	4,873	130.00	89	2.40	3	0.01	9	0.01
11	4,655	120.00	87	2.40	1	0.01	11	0.02
12	3,341	89.00	39	1.10	1	0.01	11	0.02
13	$> 114{,}400$	**	275	17.00	1	0.04	104	0.09
14	$> 127{,}100$	**	275	16.00	1	0.04	104	0.09
15	$> 167{,}500$	**	275	16.00	1	0.04	104	0.09
16	$> 134{,}200$	**	479	29.00	1	0.05	124	0.13
17	$> 90{,}500$	**	479	30.00	1	0.05	108	0.12
18	$> 71{,}200$	**	479	28.00	1	0.05	150	0.15
19	$> 32{,}400$	**	19,427	3,200.00	5	2.10	383	0.50
20	$> 53{,}700$	**	19,427	3,100.00	5	2.00	336	0.46
21	$> 49{,}300$	**	19,427	3,100.00	5	2.10	377	0.48
22	$> 96{,}100$	**	875	65.00	3	0.06	0	0.02
23	$> 64{,}400$	**	855	64.00	1	0.05	0	0.01
24	$> 103{,}200$	**	45	3.50	1	0.05	0	0.01
25	$> 90{,}100$	**	7,667	600.00	3	0.07	10	0.02
26	$> 33{,}800$	**	7,125	560.00	1	0.07	19	0.03
27	$> 47{,}700$	**	631	50.00	1	0.07	17	0.03
28	$> 95{,}600$	**	$> 95{,}100$	**	3	0.10	1	0.02
29	$> 98{,}000$	**	$> 83{,}200$	**	1	0.10	1	0.03
30	$> 86{,}700$	**	79	7.40	1	0.10	1	0.02
31	$> 34{,}700$	**	$> 37{,}900$	**	3	0.24	288	0.45
32	$> 34{,}500$	**	$> 38{,}300$	**	1	0.24	233	0.40
33	$> 31{,}800$	**	2,505	570.00	3	1.70	294	0.50
34	$> 27{,}900$	**	$> 35{,}100$	**	255	120.00	678	1.00
35	$> 15{,}700$	**	$> 35{,}300$	**	151	70.00	857	1.30
36	$> 15{,}300$	**	$> 33{,}500$	**	27	17.00	31,199	46.00
37	$> 9{,}400$	**	$> 11{,}700$	**	3	0.62	804	2.00
38	$> 10{,}100$	**	$> 11{,}600$	**	1	0.61	958	2.20
39	$> 10{,}500$	**	$> 11{,}300$	**	1	0.84	900	2.30

** Time overflow ($> 8{,}000.00$ seconds)

reduction tests such as those presented for *Steiner* problems in graphs [40, 21] and hierarchical network design problems [20]. Future work might explore these issues and also the development of polynomial time complexity heuristics perhaps with theoretical performance guarantees similar to those presented in [8] for the multi-level network design problems which are closely related to the MLNO problem treated in this paper. Finally, with the emerging emphasis on topological robustness and reliability [6, 29], it is worth mentioning the importance of the study of enhanced models with connectivity constraints which is a very promising area for future research.

Acknowledgment

The authors wish to thank Prof. Nelson Maculan for his careful reading and valuable comments on an earlier version of this paper, as well as two anonymous referees for their constructive suggestions. The research of Frederico Rodrigues Borges da Cruz and Geraldo Robson Mateus has been financed in part by the Brazilian agencies *Conselho Nacional de Desenvolvimento Científico e Tecnológico*, CNPq, and *Fundação de Amparo à Pesquisa do Estado de Minas Gerais*, FAPEMIG.

References

[1] E. Aarts and J. Korst. *Simulated Annealing & Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. John Wiley & Sons, New York, 1989.

[2] A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.

[3] K. Altinkemer and B. Gavish. Heuristics with constant error guarantees for the design of tree networks. *Management Science*, 32:331–341, 1988.

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

[5] A. Balakrishnan and K. Altinkemer. Using a hop-constrained model to generate alternative communication network design. *ORSA Journal on Computing*, 4:192–205, 1992.

[6] A. Balakrishnan, T. L. Magnanti, and P. Mirchandani. Designing hierarchical survivable networks. Working Paper OR 291-94, Sloan School of Management, MIT, 1994.

[7] A. Balakrishnan, T. L. Magnanti, and P. Mirchandani. A dual-based algorithm for multi-level network design. *Management Science*, 40(7):567–581, 1994.

[8] A. Balakrishnan, T. L. Magnanti, and P. Mirchandani. Modeling and heuristic worst-case performance analysis of two-level network design problem. *Management Science*, 40(7):846–867, 1994.

[9] A. Balakrishnan, T. L. Magnanti, A. Shulman, and R. T. Wong. Models for planning capacity expansion in local access telecommunication networks. *Annals of Operations Research*, 33:239–284, 1991.

[10] J. Barcelo and J. Casanovas. A heuristic Lagrangean algorithm for the capacitated plant location problem. *European Journal of Operational Research*, 15:212–226, 1984.

[11] T. Barrera, J. Griffith, G. Robins, and T. Zhang. Closing the gap: Near-optimal Steiner trees in polynomial time. Technical Report CS-93-31, Department of Computer Science, University of Virginia, Charlottesville, USA, 1993.

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

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

[14] N. Christofides and J. E. Beasley. A tree search algorithm for the p-median problem. *European Journal of Operational Research*, 10:196–204, 1982.

[15] CPLEX Optimization, Inc. *Using the CPLEX™ Callable Library and CPLEX™ Mixed Integer Library*. CPLEX Optimization, Inc., Incline Village, NV, 1993.

[16] 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.

[17] F. R. B. Cruz, J. MacGregor Smith, and G. R. Mateus. Solving to optimality the uncapacitated fixed-charge network flow problem. *Computers & Operations Research*, 25(1):67–81, 1998.

[18] F. R. B. Cruz, G. R. Mateus, and H. P. L. Luna. Algorithm for hierarchical network design. In *TIMS/ORSA Joint National Meeting*, page 64, Nashville, USA, 1991.

[19] J. R. Current, C. S. ReVelle, and J. L. Cohon. The hierarchical network design problem. *European Journal of Operational Research*, 27:57–66, 1986.

[20] C. W. Duin and A. Volgenant. Reducing the hierarchical network design problem. *European Journal of Operational Research*, 39:332–344, 1989.

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

[22] D. Erlenkotter. A dual-based procedure for uncapacitated facility location. *Operations Research*, 26(6):992–1009, 1978.

[23] M. L. Fisher. The Lagrangean relaxation method for solving integer programming problems. *Management Science*, 27:1–18, 1981.

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

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

[26] 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.

[27] B. Gavish. Topological design of telecommunication networks - Local access design methods. *Annals of Operations Research*, 33:17–71, 1991.

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

[29] M. Gendreau, M. Labbé, and G. Laporte. Efficient heuristics for the design of ring networks. *Telecommunication Systems*, 4(3-4):177–188, 1995.

[30] A. M. Geoffrion and G. W. Graves. Multicommodity distribution system design by Benders decomposition. *Management Science*, 20:822–844, 1974.

[31] F. Glover. Tabu search - Part I. *ORSA Journal on Computing*, 1:190–206, 1989.

[32] F. Glover. Tabu search - Part II. *ORSA Journal on Computing*, 2:4–32, 1990.

[33] M. X. Goemans. The Steiner tree polytope and related polyhedra. *Mathematical Programming*, 63:157–182, 1994.

[34] M. X. Goemans, A. V. Goldeberg, S. Plotkin, D. Shmoys, É. Tardos, and D. P. Williamson. Improved approximation algorithms for network design problems. In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, 1994.

[35] M. Held and R. M. Karp. The traveling salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.

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

[37] P. N. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner trees. In *Proc. 3rd Symposium on Integer Programming and Combinatorial Optimization*, pages 323–332, 1993.

[38] 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.

[39] N. Maculan. The Steiner problem in graphs. *Annals of Discrete Mathematics*, 31:185–212, 1987.

[40] 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.

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

[42] 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.

[43] G. R. Mateus and H. P. L. Luna. Combinatorial optimization in telephonic network planning. In *Workshop on Practical Combinatorial Optimization*, pages 40–54, Rio de Janeiro, Brazil, 1989. IFORS/ALIO.

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

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

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