Algorithms for a Multi-level Network Optimization Problem

F. R. B. Cruz\textsuperscript{a,}\textsuperscript{*}, J. MacGregor Smith\textsuperscript{b}, G. R. Mateus\textsuperscript{c}

\textsuperscript{a}Departamento de Estatística, \\
\textsuperscript{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

\textsuperscript{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.

\textsuperscript{*}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 $D = (N, A)$ 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.

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. $M^l = \sum_{L=1}^m \sum_{i \in D^l} d_i$;

$s^l$ - 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):

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

s.t.:

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l = -\left( \sum_{j \in \delta^+(i)} x_{ij}^{l-1} - \sum_{j \in \delta^-(i)} x_{ji}^{l-1} \right), \quad \forall i \in R^l,$$

$$l = 2, 3, \ldots, m, \quad (2)$$

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l = 0, \quad \forall i \in T^l,$$

$$l = 1, 2, \ldots, m, \quad (3)$$
\[
\begin{align*}
\sum_{j \in \Delta^+(i)} x_{ij}^l - \sum_{j \in \Delta^-(i)} x_{ji}^l &= -d_i, \quad \forall \ i \in D^l, \quad l = 1, 2, \ldots, m, \quad (4) \\
\sum_{j \in \Delta^+(i)} x_{ij}^l - \sum_{j \in \Delta^-(i)} x_{ji}^l &\leq s^l z_i, \quad \forall \ i \in R^l, \quad l = 1, 2, \ldots, m, \quad (5) \\
x_{ij}^l &\leq M^l y_{ij}^l, \quad \forall \ (i, j) \in A, \quad l = 1, 2, \ldots, m, \quad (6) \\
x_{ij}^l &\geq 0, \quad \forall \ (i, j) \in A, \quad l = 1, 2, \ldots, m, \quad (7) \\
y_{ij}^l &\in \{0, 1\}, \quad \forall \ (i, j) \in A, \quad l = 1, 2, \ldots, m, \quad (8) \\
z_i &\in \{0, 1\}, \quad \forall \ i \in R^l, \quad l = 1, 2, \ldots, m. \quad (9)
\end{align*}
\]

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 \(c_{ij}\) 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, $U_{\text{BEST}}$ is the global upper bound and $\Gamma$ is a list of unexplored problems $(M)^i$, each of which is of the form $Z^i_M = \min \{ c\mathbf{x} \text{ s.t.: } \mathbf{x} \in S^i \}$, 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 $\Gamma$ are a lower bound $L^i \leq Z^i_M$ and an upper bound $U^i \geq Z^i_M$. 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**

```plaintext
algorithm Branch-and-Bound
    $U_{\text{BEST}} \leftarrow +\infty$
    $\Gamma \leftarrow \{(M)^0\}$
    while $\Gamma \neq \emptyset$ do
        /* search rule */
        select and delete a problem $(M)^i$ from $\Gamma$
        /* bound rule */
        Compute_Lower_and_Upper_Bounds($L^i, U^i$)
        update $U_{\text{BEST}}$
        /* branch rule */
        Reduce[$(M)^i$]
        if $L^i < U_{\text{BEST}}$ and $(M)^i$ is not a leaf then
            $\Gamma \leftarrow \Gamma \cup \{(M)^{2i+1}\} \cup \{(M)^{2i+2}\}$
        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,
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 \( w_{ij}^* \geq 0 \). Then, the Lagrangean function is:

\[
\mathcal{L}(x, y, z; v, w^1, w^2, \ldots, w^m) = \sum_{i=1}^{m} \left[ \sum_{(i,j) \in A} \left( c_{ij}^1 x_{ij}^1 + f_{ij}^1 y_{ij}^1 \right) + \sum_{i \in R} f_i z_i \right] + \\
\sum_{i=1}^{m} \sum_{i \in R^c \cap J} v_i \left( \sum_{j \in \delta^+(i)} x_{ij}^1 - \sum_{j \in \delta^-(i)} x_{ij}^1 - s^1_i z_i \right) + \\
\sum_{i=1}^{m} \sum_{(i,j) \in K^1} w_{ij}^* \left( x_{ij}^1 - M^1 y_{ij}^1 \right),
\]

which results in the following Lagrangean relaxation:

\[
(L_{R, w^1, w^2, \ldots, w^m}): \\
L(v, w^1, w^2, \ldots, w^m) = \min_{v, w^1, w^2, \ldots, w^m \geq 0} \mathcal{L}(x, y, z; v, w^1, w^2, \ldots, w^m),
\tag{11}
\]

\text{s.t.:}

\[
(2)-(4), (7)-(9).
\]

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

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

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

\[
L(v, w^1, w^2, \ldots, w^m) = L_1(v, w^1, w^2, \ldots, w^m) + \\
L_2(v, w^1, w^2, \ldots, w^m) + \\
L_3(v, w^1, w^2, \ldots, w^m),
\tag{13}
\]

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

The subproblem \((L_1)\) is:

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

s.t.:

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

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

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

\[
\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l \leq s_i^l, \quad \forall \ i \in R^l \cap J^1, \quad l=1,2,\ldots,m, \tag{18}
\]

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

\[
x_{ij}^l \leq M_i^l, \quad \forall \ (i,j) \in K_i^1, \quad l=1,2,\ldots,m, \tag{20}
\]

\[
x_{ij}^l \leq 0, \quad \forall \ (i,j) \in K_i^0, \quad l=1,2,\ldots,m, \tag{21}
\]

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

where

\[
C_{ij}^l = \begin{cases} 
+\infty & (i,j) \in K_i^0, \\
\phi_{ij}^l & (i,j) \in K_i^1, \ j \not\in R_i^l \cap J, \ i \not\in R_i^l \cap J, \\
\phi_{ij}^l + v_i & (i,j) \in K_i^1, \ j \not\in R_i^l \cap J, \ i \in R_i^l \cap J, \\
\phi_{ij}^l - v_j & (i,j) \in K_i^1, \ j \in R_i^l \cap J, \ i \not\in R_i^l \cap J, \\
\phi_{ij}^l + v_i - v_j & (i,j) \in K_i^1, \ j \in R_i^l \cap J, \ i \in R_i^l \cap J, \\
\phi_{ij}^l + w_i j & (i,j) \in K_i^1, \ j \not\in R_i^l \cap J, \ i \not\in R_i^l \cap J, \\
\phi_{ij}^l + w_{ij}^l + v_i & (i,j) \in K_i^1, \ j \not\in R_i^l \cap J, \ i \not\in R_i^l \cap J, \\
\phi_{ij}^l + w_{ij}^l - v_j & (i,j) \in K_i^1, \ j \not\in R_i^l \cap J, \ i \not\in R_i^l \cap J, \\
\phi_{ij}^l + w_{ij}^l + v_i - v_j & (i,j) \in K_i^1, \ j \not\in R_i^l \cap J, \ i \not\in R_i^l \cap J.
\end{cases} \tag{23}
\]

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 \(K_1^1\) and \(K_2^0\), and so on for all remaining levels. The complete algorithm is seen in Figure 3.

\[
\text{Figure 3 goes around here}
\]
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:

\[
\sum_{(i,j) \in A(C^l)} C^l_{ij} = \sum_{(i,j) \in A(C^l)} e^l_{ij} + \sum_{(i,j) \in A(C^l) \cap K^l} w^l_{ij} + \sum_{i \in N(C^l) \cap R^l \cap J} v_i - \sum_{j \in N(C^l) \cap R^l \cap J} v_j
\]

\[
= \sum_{(i,j) \in A(C^l)} e^l_{ij} + \sum_{(i,j) \in A(C^l) \cap K^l} w^l_{ij}
\]
\[ \geq 0. \]

**Subproblem \((L_2)\)**

The subproblem \((L_2)\) is a subset selection problem:

\[
L_2(v, w^1, w^2, \ldots, w^m) = \min \sum_{l=1}^{m} \sum_{(i,j) \in A} (f^l_{ij} - w^l_{ij} M^l) y^l_{ij},
\]

\text{s.t.:}

\[
\begin{align*}
y^l_{ij} & \in \{0, 1\}, \quad \forall (i,j) \in K^l, \\
y^l_{ij} & = 1, \quad \forall (i,j) \in K^l_1, \\
y^l_{ij} & = 0, \quad \forall (i,j) \in K^l_0,
\end{align*}
\]

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:

\[
L_3(v, w^1, w^2, \ldots, w^m) = \min \sum_{l=1}^{m} \sum_{i \in R^l} (f_i - v_i s^l) z_i,
\]

\text{s.t.:}

\[
\begin{align*}
z_i & \in \{0, 1\}, \quad \forall i \in R^l \cap J, \\
z_i & = 1, \quad \forall i \in R^l \cap J_1, \\
z_i & = 0, \quad \forall i \in R^l \cap J_0,
\end{align*}
\]

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 \(c^l_{ij}\), the fixed costs of the used arcs \(f^l_{ij}\), 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](image)

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

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

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.

```
procedure Chosen_Arc_No_Cycling
    /* try to choose a first level candidate supply node */
    for \(\forall i \in R^1\) do
        if \(i \in J\) 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 \(\forall i \in R^{l+1}\) do
            if \(i \in J\) then
                return \(i\)
            end if
        end for
        /* otherwise try to choose an arc in the \(l\)-th level */
        /* compute set of reached nodes \(T\) */
        \(T \leftarrow \emptyset\)
        for \(\forall i \in N\) do
            if \(i \in R^l \cap J_1\) then
                \(T \leftarrow T \cup \{i\}\)
            end if
        end for
        for \(\forall (i, j) \in A\) do
            if \((i, j) \in K^l_1\) then
                \(T \leftarrow T \cup \{i\} \cup \{j\}\)
            end if
        end for
        /* search new eligible free arc */
        if there is \(i \in D^l \cup (R^{l+1} \cap J_1)\) such that \(i \notin T\) then
            for \(\forall (i, j) \in A\) do
                if \((i, j) \in K^l_1\) and \(i \in T\) and \(j \notin T\) 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, \((LR_{v,w_1,w_2,...,w_m})\), but without subgradient optimizations. The algorithm is presented in Figure 6.

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 \( 100 \times 100 \) 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 \( \Omega_{ij} \) were defined as the Euclidean distance between the extremities \( i \) and \( j \).
procedure Reduce(M)
    /* initialize set of nodes and arcs recently fixed */
    F_s ← ∅; 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_s ← F_s \ {i}
                else
                    J_0 ← J_0 \ {i}; J_1 ← J_1 ∪ {i}; F_s ← F_s \ {i}
                end if
            end for
            /* try to fix free arcs */
            for ∀(i, j) ∈ K^l do
                K^l ← K^l \ {(i, j)}; K^l_1 ← K^l_1 \ {(i, j)}
                if L(v, w^1, w^2, . . . , w^m) > U_{BEST} then
                    K^l_1 ← K^l_1 \ {(i, j)}; K^l_0 ← K^l_0 \ {(i, j)}; F_K^l ← F_K^l \ {(i, j)}
                else
                    K^l_0 ← K^l_0 \ {(i, j)}; K^l_1 ← K^l_1 \ {(i, j)}; F_K^l ← F_K^l \ {(i, j)}
                end if
            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, . . . , m, with the respective cardinalities, each node being chosen at most once. All demands were considered unitary. The costs f^l_{ij} and c^l_{ij} 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^1_{ij}, c^1_{ij}, and f_s 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 = $100\% \times \frac{(U_{BEST} - L^0)}{U_{BEST}}$, 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 $f_1$, 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

| Problem | $|N|$ | $|A|$ | $|R^1|$ | $|D^1|$ | $|R^2|$ | $|D^2|$ | $\frac{1}{N}$ | $\frac{1}{M}$ | $\frac{2}{N}$ | $\frac{2}{M}$ | $f_i$ | $U_{\text{BEST}}$ | GAP(%) | CPU(s) | $U_{\text{opt}}$ |
|---------|-----|------|--------|--------|--------|--------|--------|--------|--------|--------|--------|------|----------|------|--------|----------|
| 1       | 4   | 6    | 1      | 1      | 1      | 1      | 8      | 2      | 12     | 8      | 2      | 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      | 12     | 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   | 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      | 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     | 2      | 4      | 1      | 8      | 2      | 128   | 0     | 17,724| 1.60  | 0.37   | 17,724   |
| 20      | 8   | 17,748| 1.60   | 0.38   | 17,748 |
| 21      | 1024| 20,796| 6.30   | 0.38   | 20,796 |
| 22      | 16  | 30    | 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     | 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   | 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     | 2      | 4      | 1      | 8      | 2      | 128   | 0     | 16,669| 1.20  | 1.70   | 16,669   |
| 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>
<thead>
<tr>
<th>Problem</th>
<th>Branch-and-Bound</th>
<th>CPLEX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Chosen_Arc_First_Free</td>
<td>Chosen_Arc_No_Cycling</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>6,655</td>
<td>45</td>
</tr>
<tr>
<td>8</td>
<td>6,655</td>
<td>45</td>
</tr>
<tr>
<td>9</td>
<td>6,655</td>
<td>45</td>
</tr>
<tr>
<td>10</td>
<td>4,873</td>
<td>89</td>
</tr>
<tr>
<td>11</td>
<td>4,655</td>
<td>87</td>
</tr>
<tr>
<td>12</td>
<td>3,341</td>
<td>39</td>
</tr>
<tr>
<td>13</td>
<td>&gt; 114,400</td>
<td>**</td>
</tr>
<tr>
<td>14</td>
<td>&gt; 127,100</td>
<td>**</td>
</tr>
<tr>
<td>15</td>
<td>&gt; 167,500</td>
<td>**</td>
</tr>
<tr>
<td>16</td>
<td>&gt; 134,200</td>
<td>**</td>
</tr>
<tr>
<td>17</td>
<td>&gt; 90,500</td>
<td>**</td>
</tr>
<tr>
<td>18</td>
<td>&gt; 71,200</td>
<td>**</td>
</tr>
<tr>
<td>19</td>
<td>&gt; 32,400</td>
<td>**</td>
</tr>
<tr>
<td>20</td>
<td>&gt; 53,700</td>
<td>**</td>
</tr>
<tr>
<td>21</td>
<td>&gt; 49,300</td>
<td>**</td>
</tr>
<tr>
<td>22</td>
<td>&gt; 96,100</td>
<td>**</td>
</tr>
<tr>
<td>23</td>
<td>&gt; 64,400</td>
<td>**</td>
</tr>
<tr>
<td>24</td>
<td>&gt; 103,200</td>
<td>**</td>
</tr>
<tr>
<td>25</td>
<td>&gt; 90,100</td>
<td>**</td>
</tr>
<tr>
<td>26</td>
<td>&gt; 33,800</td>
<td>**</td>
</tr>
<tr>
<td>27</td>
<td>&gt; 47,700</td>
<td>**</td>
</tr>
<tr>
<td>28</td>
<td>&gt; 95,600</td>
<td>**</td>
</tr>
<tr>
<td>29</td>
<td>&gt; 98,000</td>
<td>**</td>
</tr>
<tr>
<td>30</td>
<td>&gt; 86,700</td>
<td>**</td>
</tr>
<tr>
<td>31</td>
<td>&gt; 34,700</td>
<td>**</td>
</tr>
<tr>
<td>32</td>
<td>&gt; 34,500</td>
<td>**</td>
</tr>
<tr>
<td>33</td>
<td>&gt; 31,800</td>
<td>**</td>
</tr>
<tr>
<td>34</td>
<td>&gt; 27,900</td>
<td>**</td>
</tr>
<tr>
<td>35</td>
<td>&gt; 15,700</td>
<td>**</td>
</tr>
<tr>
<td>36</td>
<td>&gt; 15,300</td>
<td>**</td>
</tr>
<tr>
<td>37</td>
<td>&gt; 9,400</td>
<td>**</td>
</tr>
<tr>
<td>38</td>
<td>&gt; 10,100</td>
<td>**</td>
</tr>
<tr>
<td>39</td>
<td>&gt; 10,500</td>
<td>**</td>
</tr>
</tbody>
</table>

** 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


