PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Reporte Técnico RT 15-11

The capacitated *m* two node survivable star problem

Gabriel Bayá, Antonio Mauttone

Franco Robledo

2015

The capacitated *m* two node survivable star problem

Gabriel Baryá, Antonio Mauttone, Franco Robledo

ISSN 0797-6410
Reporte Técnico RT 15-11
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay, 2015

The Capacitated m Two Node Survivable Star Problem

Gabriel Bayá[a], Antonio Mauttone[b] and Franco Robledo[b]

[a]*Master Program on Informatics*

[b]*Departamento de Investigación Operativa , Instituto de Computación*

Facultad de Ingeniería, Universidad de la República.

Julio Herrera y Reissig 565, Montevideo, Uruguay.

Emails: {gbaya,mauttone,frobledo}@fing.edu.uy

Abstract. The problem addressed in this paper attempts to efficiently solve a network design with redundant connections, often used by telephone operators and internet services. This network connects customers with one master node and sets some rules that shape its construction, such as number of customers, number of components and types of links, in order to meet operational needs and technical constraints. We propose a combinatorial optimization problem called CmTNSSP (Capacitated m Two-Node-Survivable Star Problem), a relaxation of CmRSP (Capacitated m Ring Star Problem). In this variant of CmRSP the rings are not constrained to be cycles; instead, they can be two node connected components. The contributions of this paper are (a) introduction and definition of a new problem (b) the specification of a mathematical programming model of the problem to be treated, and (c) the approximate resolution thereof through a GRASP metaheuristic, which alternates local searches that obtain incrementally better solutions, and exact resolution local searches based on mathematical programming models, particularly Integer Linear Programming ones. Computational results obtained by developed algorithms show robustness and competitiveness when compared to results of the literature relative to benchmark instances. Likewise, the experiments show the relevance of considering the specific variant of the problem studied in this work.

Keywords: Topological Network Design, Survivability, Greedy Randomized Adaptive Search Procedure (GRASP), Variable Neighborhood Search (VNS), Metaheuristics.

1 Introduction

Along with the evolution of telephone communications, it began the development of computers and digital data transmission. To communicate two remote computers the telephone network was used as a transmission medium. This fact generated a number of associated services that settled in a communications infrastructure that had grown up without sufficient planning. Therefore some events occurred which devastating consequences are directly linked to this lack of planning, such as the burning of a telephone exchange in a suburb of Chicago in May 1988, which rendered uncommunicated 35,000 local subscribers, and affected 120,000 long distance trunk lines, compromising the functioning at O'Hare air traffic control and outaging 911 service, as detailed in the report *Keeping the Phone Lines Open* by Zorpette (1989). These accidents reveal, among other things, the need for proper planning of telephone networks and data transmission. Beyond all preventive actions that can be taken to avoid accidents as the one quoted above, a key element to mitigate such impact is a proper design of telecommunication networks. The study of the structure, the introduction of minimum levels of connectivity between their nodes, and redundancy are crucial to avoid catastrophic events upon the occurrence of a failure. The main motivation for studying topological network design is its application in the area of telecommunications (Stoer, 1992). Basically, the goal is to obtain structures with the desired level of redundancy and fault-tolerant in some of its nodes or their links, and to allow savings in construction costs. Initially, topological network design covered mainly availability aspects (e.g. public switched telephone network). However, new applications over the Internet infrastructure reveal the shortcomings of tree-like structures. On the other hand, mesh-like structures present valuable connectivity properties but their deployment is prohibitively expensive. A natural approach to an acceptable level of connectivity is to connect all terminals in a ring or cycle in the cheapest way. This problem is known as Traveling Salesman Problem Dantzig and Fulkerson (1954) and it is widely studied in the scientific literature. In the physical design of a telephony deployment, it is useful to consider several two-connected components joined to a perfect telephone exchange, but if some terminal nodes are far away, it is better to connect them in more than one ring. A cost-effective "shape" of a solution is provided by Baldacci et al. (2007). In that work, given a *depot*, several terminal nodes and optional nodes, in order to connect all terminals, the authors propose to find the cheapest m rings joined in the depot, while some terminals can be pending on some node of a ring. The number of nodes within a ring must not exceed the depot capacity, and the cost of pending nodes is different than the cost of the connections within the rings. The minimum-cost design of the m-rings is called Capacitated m Ring Star Problem, termed here CmRSP for short. Furthermore a cornerstone in the area of topological network design was offered by Monma et al. (1990). The authors fully characterize the structure of minimum-cost two-node connected sub-networks in metric graphs. They proved that a minimum-cost two-node connected metric network is either a Hamiltonian tour or presents a special graph topology as an induced sub-graph sketched in Figure 1. Motivated by this result we studied in a problem with two-node-connected structures that can potentially have better cost than cycles.

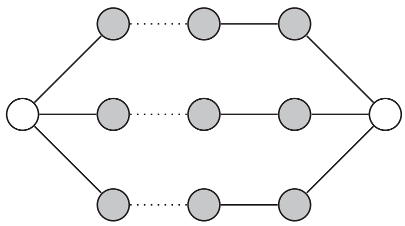

Figure 1: Monma's graph structure.

In the literature we have not found references to *Capacitated m Two-Node-Survivable Star Problem* as is. Related work developed by Baldacci et al. (2007) treat exact resolution of *Capacitated m Ring Star Problem*. Such problem is slightly different to the problem treated in this paper. In CmRSP 2-node-connected structures are exclusively cycles, whereas in our problem (CmTNSSP) other two-node-connected structures are allowed. The CmTNSSP is therefore a CmRSP relaxation. Baldacci et al. (2007) consider two mathematical programming formulations of CmRSP, which is solved exactly by comparing the result of the implementation of both formulations for a set of test cases; the authors propose a set of test instances comprising up to 100 nodes. Some authors also treat CmRSP and solve it exactly (Hoshino and De Souza, 2012) and other authors do it using approximate methods. For example we can cite Naji-Azimi et al. (2010) who use iterated heuristics and Mauttone et al. (2008) who use the GRASP metaheuristic. Moreover, Naji-Azimi et al. (2012) proposed heuristics based on integer linear programming (ILP) for the CmRSP and also they proposed larger instances comprising up to 200 nodes. More recently, Zhang et al. (2014) proposed a memetic algorithm which improves previous results and also the authors explore new instances comprising new cost structures.

There are some works that underly Baldacci's work such *Locating Median Cycles in Networks*. This problem is a particular case of CmRSP and it is studied by Labbé et al. (2005). In this problem the authors seek to build a network that consists of a main loop and nodes attached to it, whose total cost is minimum. The cost is composed by cost of the edges that belong to the cycle (routing costs) plus costs of connection of the edges with incidence in attached nodes. Here, the total connection cost is bounded to a value given. In Labbé et al. (2004) the same authors solve the RSP (Ring Star Problem), without imposing cost constraints on the edges that do not belong to the cycle. Only service constraint are mentioned in this problem such as number of attached nodes connected to the same node belonging to the cycle. In that study also RSP is solved exactly. Other similar problems but with differences in the structures are discussed in Richey (1990). In the CmRSP and in the CmTNSSP (the problem addressed in this paper), the structure of feasible solutions are cycles or two-connected structures, while in the other problems mentioned above they are simple connected structures without redundancy such as paths or trees.

In this paper we propose an alternative (to best our knowledge not yet studied) to design 2-node-connected low-cost solutions, useful in the context of building telecommunications networks with some level of survivability. We define the CmTNSSP and propose an ILP model to solve exactly small instances. Also, we propose and implement an hybrid metaheuristic which is then applied to known instances of the literature, and to other tests cases specifically designed. This article is organized as follows. Description and formal definition are presented in Section 2. Integer lineal programming model is presented in Section 3. A GRASP-VND metaheuristic is developed for the approximated resolution in Section 4. Computational results are reported in Section 5. Conclusions and trends for future work are discussed in Section 6.

2 Problem Definition

The problem to be described is an example about planning that must be followed to build fault-tolerant networks that meet some operational needs and technical constraints.

2.1 Problem description

Given a simple non directed graph $G = (V, E)$, we want to get a sub-graph (network) that meets certain topology, formally defined in Section 2.2. In this graph G we have a distinguished node "d" that we call depot. Across the scope of this work the term " node " is used to refer to any vertex within the set of vertices of any of the defined graphs. Both terms will be used interchangeably. The set of remaining vertices $V \setminus \{d\}$ will be partitioned into two disjoint sets, one called set of terminal nodes T, and the other called set of auxiliary or Steiner nodes W. Terminal nodes are those that must be necessarily present in the network and auxiliary ones participate in the solution only if its inclusion improves construction costs of such network.

A feasible solution consists of a certain number m of related sub-graphs, which will share the d node, so that if we remove this node the resulting graph would be divided into m connected-components. Each component connects depot d with a set of terminal nodes which cardinality cannot exceed a given capacity Q. This parameter narrows the number of nodes of each component in response to connection constraints and latency in communications. Terminal nodes present in each of these m connected-components either belong to an associated structure with redundancy which is part of the component, or are attached to such structure by an edge. In this associated structure with redundancy, every pair of vertices are connected by two independent paths. Steiner nodes, if included, can belong to redundancy associated structures but cannot be attached to these structures by an edge.

The graph G has two associated matrix costs. One of them determines the cost of connecting each pair of vertices if both are part of the related structure with redundancy (routing costs) and the other determines the cost of connecting a pair of vertices if one of them is attached to the structure by an edge (connection costs). Usually when designing networks the cost of the core routers is greater than the cost of access routers, therefore this situation is covered by the definition of different costs.

Our problem consist in getting a sub-graph of G, which is of minimum cost and built under the above assumptions. We will call this problem *Capacitated m Two Node Survivable Star Problem* (CmTNSSP). In Figure 2 we can see an example of a feasible solution, where the rectangular node is the depot, black nodes are terminals and the white node is optional. Edges drawn with full lines describe routing costs and dotted ones connection costs.

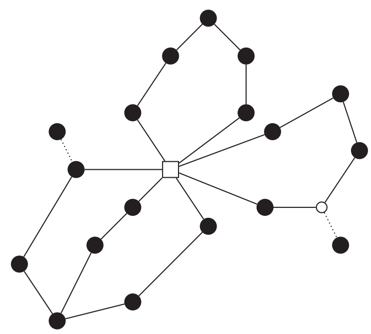

Figure 2: An example of CmTNSSP solution.

2.2 Formal definition

To give a formal definition of CmTNSSP we will establish some definitions and conventions which we will work hereinafter.

Network design problems with connectivity requirements can be defined in two ways:

- With respect to the amount of edges (links) that may fail in the network without leaving disconnected any two terminal nodes. These requirements translate into edge-disjoint paths between pairs of terminal nodes.

- With respect to the number of nodes that can fail (together with their incident edges) without leaving disconnected any two terminal nodes. These requirements result in node-disjoint paths between pairs of terminal nodes.

The following definitions are taken from Stoer (1992).

Definition 1. *Let $G = (V, E)$ an undirected graph with V a set of vertices and E a set of edges, a pair of nodes $(i, j) \in V \times V$ has k-edge-connectivity or is k-edge-connected in G, when at least k edge-disjoint paths (which share no edge) are connecting i with j.*

This definition is equivalent to stating that any cut in the graph for nodes i, j contains at least k edges.

Definition 2. *We say that a graph $G = (V, E)$ is k-edge-connected if, for every pair of nodes $(i, j)\ inV$, this couple is k-edge-connected.*

Analogously the node-connectivity concepts are defined.

Definition 3. *We say that a pair of nodes (i, j) has k-node-connectivity or is k-node-connected in a given graph, when at least k node-disjoint paths (i.e. they do not share any nodes except i , j) are connecting i with j.*

Definition 4. *We say that a graph is k-node-connected if every pair of nodes i, j thereof is k-node-connected.*

Readers can note that if two paths with the same endpoints i, j are node-disjoint, then they are also edge-disjoint; but not reciprocally.

Definition 5. *Given a graph $G = (V, E)$ and a vertex $i \in V$ we call degree of i and we noted $\delta(i)$ to the number of incident edges to node i.*

Once specified these definitions, let us now turn to the formal definition of CmTNSSP.

Let $G = (V, E)$ be a graph where V is a set of vertices, E is a set of edges and $d \in V$ is a distinguished node of V called $depot$.
Let $T \subseteq V \setminus \{d\}$ a set of nodes, which we call terminal nodes of the graph G.
Let $\hat{T} = T \cup \{d\}$ the set of terminals with $depot$ node included.
Let $W = V \setminus \hat{T}$ a set of optional (or Steiner nodes) of G.

We want to construct a graph H in such a way that:

$$H = H_1 \cup H_2 \cup H_3 \cdots\cdots \cup H_m \tag{1}$$

where each component H_i is defined as:

$$H_i = G'_i \cup S_i \quad i = 1, \cdots m \tag{2}$$

and meets:

- $G'_i = (U'_i, E'_i) \qquad U'_i \subseteq V$ y $E'_i \subseteq E, \qquad i = 1, \cdots m$ are 2-node-connected graphs,
- $S_i = (\bar{V}_i \cup \bar{U}_i, \bar{E}_i), \quad \bar{U}_i \subseteq U'_i, \quad \bar{V}_i \subset T, \quad \bar{V}_i \cap U'_i = \phi,$
  $\bar{E}_i = \{(u_i, v_i)\}, \quad u_i \in \bar{U}_i, \quad v_i \in \bar{V}_i, \quad \bar{E}_i \subset E$
  $\delta(v_i) = 1 \quad \forall v_i \in \bar{V}_i \qquad i = 1, \cdots m$.

Hereinafter the set of nodes $v_i \in \bar{V}_i$ we will call pendant nodes, the set of nodes $u_i \in \bar{U}_i$ base of pendant nodes, and the set of edges $\{(u_i, v_i)\} \in \bar{E}_i$ we will call pendant edges. Let $T(H_i)$ the set of terminal nodes of the i-th component of the graph H, there is a capacity constraint such that:

$$|T(H_i)| \leq Q \tag{3}$$

For the distinguished node d defined above, the following is met:

$$d = H_1 \cap H_2 \cap H_3 ...... \cap H_m \tag{4}$$

We also define $C = \{c_{ij}\}_{i,j \in V}$ as the routing costs, i.e. the cost of a certain edge (i, j) which belongs to some G'_k, with $k = 1 \cdots m$. Analogous, let us now define $D = \{d_{ij}\}_{i,j \in V}$ as the connection costs matrix, i.e. the cost of the edge (i, j) when this edge belongs to S_k, with $k = 1 \cdots m$.

Our goal is to construct a graph H defined above, which should be a minimum cost graph, where such cost includes routing and connection costs.

Proposition 1. ***(Complexity)*** *CmTNSSP belongs to class of $\mathcal{NP}$-Hard problems.*

Proof. Given an undirected graph $G = (V, E)$ the Minimum-Weight Two-Connected Spanning Network (Monma et al., 1990) is a particular case of CmTNSSP with $m = 1, \quad Q = |V|$, $W = \phi, \quad$ y $\bar{V}_1 = \phi$. Last condition can be forced making large enough elements of connection costs matrix D. As the Minimum-Weight Spanning Two-Connected Network belongs to the class of problem $\mathcal{NP}$-Hard (Monma et al., 1990) this demonstrates CmTNSSP also belongs to the same class.

□

3 Integer Linear Programming model

In this section we propose an integer lineal programming model for the CmTNSSP. This model was translated to an algebraic language and solved as we will see in Section 5.
Let be the set of adjacent nodes to node $i \in V$ as $Adj(i)$ as follow:

$$Adj(i) = \{j \in V : (i, j) \in E\}$$

Below we define decision variables of the model:

$$X_i^k = \begin{cases} 1 & \text{if node } i \in V \text{ belongs to } G'_k \text{ (2-connected structure of sub-network } H_k) \\ 0 & \text{otherwise} \end{cases}$$

$$Y_i^k = \begin{cases} 1 & \text{if node } i \in T \text{ is a pendant node of } G'_k \text{ (2-connected structure of sub-network } H_k\text{)} \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{i,j}^k = \begin{cases} 1 & \text{if } i \in T \text{ and } j \in V \text{ are connected by edge } (i,j) \in E, \\ & \text{being } i \text{ a pendant node of } G'_k \text{ (2-connected structure of sub-network } H_k\text{)} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{(i,j)}^{(u,v,k)} = \begin{cases} 1 & \text{if edge } (i,j) \text{ is used in the path from } u \text{ to } v \\ & \quad \text{in the direction from } i \text{ to } j \text{ within component } H_k \\ 0 & \text{otherwise} \end{cases}$$

$$X_{i,j}^k = \begin{cases} 1 & \text{if there is a path between } i \text{ and } j \text{ within component } H_k \\ 0 & \text{otherwise} \end{cases}$$

$$x_{i,j} = \begin{cases} 1 & \text{if edge } (i,j) \text{ is used in the solution} \\ 0 & \text{otherwise} \end{cases}$$

$$w_{i,j} = \begin{cases} 1 & \text{if edge } (i,j) \text{ is a pendant edge used in the solution} \\ 0 & \text{otherwise} \end{cases}$$

$$z_i = \begin{cases} 1 & \text{if pendant node } i \text{ is used in the solution} \\ 0 & \text{otherwise} \end{cases}$$

In the following we present the mathematical programming formulation splitted in several parts representing different aspects of the problem.

$$min \sum_{k=1}^{m} \left(\sum_{i,j \in V} c_{ij}(x_{ij} - w_{ij}) + \sum_{i,j \in V} d_{ij} w_{ij} \right) \tag{5}$$

subject to:

$$\sum_{k=1}^{m} X_i^k + Y_i^k = 1 \qquad \forall i \in T \tag{6}$$

$$\sum_{k=1}^{m} X_d^k = m \tag{7}$$

$$\sum_{k=1}^{m} X_i^k \leq 1 \qquad \forall i \in W \tag{8}$$

$$Y_i^k = 0 \qquad \forall i \in W, \quad \forall k \in 1 \cdots m \tag{9}$$

$$\sum_{i \in T} (X_i^k + Y_i^k) \leq Q \qquad \forall k \in 1 \cdots m \tag{10}$$

$$Y_{ij}^k \leq x_{ij} \qquad \forall i \in T, \ \forall j \in V, \ \forall k \in 1 \cdots m \tag{11}$$

$$Y_i^k = \sum_{j \in Adj(i)} Y_{ij}^k \qquad \forall i \in T, \quad \forall k \in 1 \cdots m \tag{12}$$

Constraint 6 ensures that any terminal node, either belongs to a 2-connected structure of an unique component H_k or hangs on it, distinguished node d (depot) belongs to the m components (Constraint 7). Inequalities 8 and 9 ensure that Steiner nodes belong to the 2-connected structures of the components H_k or they don't participate in the solution, while Constraint 10 is the capacity constraint, stating that each component H_k can have Q terminal nodes excluding the d node. Constraint 11 implies that if nodes i and j are connected in the component H_k where i is a pendant node, then the edge (i, j) belongs to the solution. Inequality 12 rules that if node i hangs from the 2-connected structure, then it does across one and only one edge.

$$\sum_{(u,j) \in E} y_{(u,j)}^{(u,v,k)} \geq 2X_{(u,v)}^k - Y_u^k \qquad \forall u, v \in \hat{T}, \quad u \neq v, \quad \forall k \in 1 \cdots m \tag{13}$$

$$\sum_{(i,v) \in E} y_{(i,v)}^{(u,v,k)} \geq 2X_{(u,v)}^k - Y_v^k \qquad \forall u, v \in \hat{T}, \quad v \neq u, \quad \forall k \in 1 \cdots m \tag{14}$$

$$\sum_{(i,p) \in E} y_{i,p}^{(u,v,k)} - \sum_{(p,i) \in E} y_{(p,i)}^{(u,v,k)} \geq 0 \quad \forall u, v \in \hat{T}, \quad \forall p \in V \setminus u, v, \quad \forall k \in 1 \cdots m \tag{15}$$

$$y_{(i,j)}^{(u,v,k)} + y_{(j,i)}^{(u,v,k)} \leq x_{i,j} \quad \forall u, v \in \hat{T}, \quad u \neq v, \quad \forall (i,j) \in E, \ \forall k \in 1 \cdots m \tag{16}$$

Constraint 13 to 16 are the flow inequalities that ensure 2-connectivity between any pair of nodes on the 2-connected structure in each component and simple connectivity when one of them (or both) are pendant nodes.

$$X_i^k + X_j^k \leq 1 + X_{i,j}^k \qquad \forall i \in V, \quad \forall j \in V, \quad \forall k \in 1 \cdots m \tag{17}$$

$$X_i^k + Y_j^k \leq 1 + X_{i,j}^k \qquad \forall i \in V, \quad \forall j \in T, \quad \forall k \in 1 \cdots m \tag{18}$$

$$Y_i^k + Y_j^k \leq 1 + X_{i,j}^k \qquad \forall i \in T, \quad \forall j \in T, \quad \forall k \in 1 \cdots m \tag{19}$$

Constraints 17 to 19 indicate that if the nodes are present in the solution in the same component H_k, then they are connected.

$$2X_{i,j}^k \leq X_i^k + X_j^k + Y_i^k + Y_j^k \qquad \forall i \in V, \quad \forall j \in V, \quad \forall k \in 1 \cdots m \tag{20}$$

$$\sum_{k=1}^{m} X_{i,j}^k \leq 1 \qquad \forall i, j \in V \tag{21}$$

Constraint 20 determines that if i and j are connected within the same component H_k then it holds that such nodes belong to the solution and either both belong to the 2-connected structure of the component H_k or one of them hangs of it. Inequality 21 ensures that if nodes i and j are connected, just do it in to an unique component.

$$\sum_{k=1}^{m} Y_i^k \leq z_i \qquad \forall i \in T \tag{22}$$

$$\sum_{j \in Adj(i)} x_{i,j} - 1 \leq M(1 - z_i) \qquad \forall i \in T \quad M \in \mathbb{Z}^+, M \geq max(\delta_i) \;\; i = 1 \cdots |V| \tag{23}$$

Constraints 22 and 23 guarantee that the pendant nodes have degree one, and others have degree less or equal than the degree of the node with the highest degree of graph G.

$$\sum_{k=1}^{m} Y_{i,j}^k = w_{i,j} \qquad \forall i \in T, \quad j \in Adj(i) \tag{24}$$

$$w_{i,j} \leq x_{i,j} \qquad \forall i \in T, \quad j \in Adj(i) \tag{25}$$

$$Y_{i,j}^k \leq X_j^k \qquad \forall i \in T, \quad \forall j \in Adj(i), \quad \forall k \in 1 \cdots m \tag{26}$$

$$\Big(\sum_{i \in Ady[j]} x_{j,i} - \sum_{i \in Ady[j]} Y_{i,j}^k\Big) \geq 2X_j^k \qquad \forall j \in V \setminus T, \quad \forall k \in 1 \cdots m \tag{27}$$

$$2y_{i,j}^{(u,v,k)} \leq X_{i,j}^k + X_{u,v}^k \quad \forall u, v \in \hat{T}, \quad \forall i, j \in V, \quad u \neq v, \quad \forall k \in 1 \cdots m \tag{28}$$

Inequalities for 24 to 28 are additional inequalities needed for technical issues.

Thus we have defined a mathematical programming model of CmTNSSP. This model is of integer linear nature with polynomial number of variables and constraints on the size of the graph. Small sized problem instances can be solved by applying this model as we will see in Section 5.

4 Grasp Resolution

Given the nature of the problem and its complexity, we will address the resolution thereof by the GRASP (Greedy Randomize Adaptative Search Procedure) metaheuristic (Feo and Resende, 1995), an iterative process used with success in telecommunications Robledo (2005). GRASP comprises two phases: Construction and Local Search. In the first phase, a feasible solution is built applying greediness (intensification) and randomization (diversification) using a RCL (Restricted Candidate List), which is used to select elements to be added to the solution. In the second phase this solution is improved exploring neighbor solutions successively. The solution found by running independently both phases several times is taken as the best solution. A complete detail of generic GRASP characteristics can be read in Resende (2009).

4.1 Construction phase

Construction Phase is the first milestone to produce a feasible solution. In our problem we need to build m 2-node-connected components having depot d as common vertex. During the Construction Phase, components will be iteratively built. We describe below the stages of such phase of GRASP.

- **Step 1**. We proceed to locate the first m terminal nodes to be included (one in each component). Algorithm 1 considers m random terminals and computes the sum of distances between them. This procedure is performed n times and the set of m nodes with the maximum sum of distances between them is choosen.

- **Step 2**. For each node of the set selected in Step 1, we consider the k node-disjoint shortest (respect to the routing costs) paths between node under consideration and the depot, whose total cost is minimal. To obtain these k node-disjoint paths that meet this condition (minimum total cost) we use the algorithm developed by Bhandari (1997). The number of paths k, is a parameter of the constructor. ($k \geq 2$). From this list of k paths we choose randomly exactly two, and include them in the solution. This process is repeated m times, once for each set of k node-disjoint paths.

Algorithm 1 Selection of m initial nodes

procedure Far
input G, C, T, m, n
$bestfar \leftarrow \phi$
$maxdistance = 0$
for i=1 **to** n **do**
    $far \leftarrow \phi$
    for i=1 **to** m **do**
        $far[i] \leftarrow$ **ExtractRandomNode**(T)
    end for
    $distance = 0$
    for i=1 **to** m-1 **do**
        for j=i+1 **to** m **do**
            $distance = distance + C_{far[i],far[j]}$
        end for
        if $distance > maxdistance$ **then**
            $bestfar \leftarrow far$
            $maxdistance = distance$
        end if
    end for
end for
return $bestfar$

- **Step 3**. We add terminal nodes that still are not part of the solution under construction. Such terminals will be incorporated into each of the components as follows:

  A terminal node which does not belong to the solution under construction is selected randomly, and is connected to the solution, generating a path to some of the m components. This operation preserves 2-node-connectivity, since adding an independent path between two nodes to a 2-node-connected graph, generates a new 2-node-connected graph (Frederickson and Ja'Ja', 1981). We chose the component which connects the node using the criterion of fewer nodes present in this component. This approach is particularly useful for balancing the number of nodes in each of the m components without losing feasibility with respect to the capacity constraint Q. In this process we try to keep a trade-off of connecting the node to an "inadequate" component as far as costs are concerned.

  To do this, we transform the component adding a virtual node v' connected to all nodes of such component by zero cost edges, and likewise assigning the value 0 to the edges present in the component to be treated. Then we define $\bar{C}_{(|V|+1)\times(|V|+1)}$ as the matrix of the transformed component.

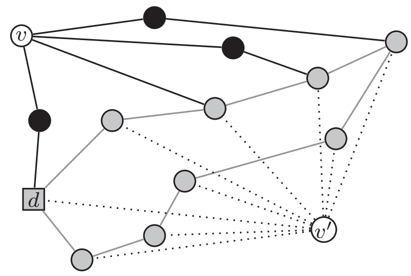

Figure 3: Including node v into a component.

Once we apply the transformation explained above we proceed to get the k node-disjoint paths with minimum total cost (again using the algorithm of Bhandari) between the terminal node to include v and the virtual node v' (see Figure 3). Of these k paths we choose any two randomly, and incorporate them in the solution under construction.

Algorithm 2 that describes the three steps that comprise the construction phase of GRASP, stops when all terminal nodes are include in some component using the procedure described above.

We remark that in construction phase the algorithm tries to build non-cyclical components using, if it improves costs, Steiner nodes. The pendant nodes are not considered at this stage and appear in the solution when local search is performed.

Algorithm 2 Construction of feasible solution

```
procedure Construct_Greedy_Randomize_Feasible_Solution
input G, C, ListSize, m, n, Q, T
G_Sol ← φ
component_nodes ← φ
not_assigned ← T
FarNodes ← Far(G, C, T, m, n)
for i=1 to m do
    node = ExtractRandomNode(FarNodes)
    minpaths = Bhandari(G, C, not_assigned, ListSize, depot, node)
    path_1 ← ExtractRandomPath(minpaths)
    path_2 ← ExtractRandomPath(minpaths)
    G_Sol ← add_path(G_Sol, path_1)
    G_Sol ← add_path(G_Sol, path_2)
    component_nodes[i] ← add_nodes(component_nodes[i], path_1)
    component_nodes[i] ← add_nodes(component_nodes[i], path_2)
    not_assigned ← subtract_nodes(not_assigned, path_1)
    not_assigned ← subtract_nodes(not_assigned, path_2)
end for
repeat
    node = ExtractRandomNode(not_assigned)
    comp = CompSelect(G_Sol)
    Ḡ = transform (G, C, C̄, G_Sol, comp, component_nodes) // Figure 3
    minpaths = Bhandari(Ḡ, C̄, not_assigned, ListSize, node, virtual)
    path_1 ← ExtractRandomPath(minpaths)
    path_2 ← ExtractRandomPath(minpaths)
    G_Sol ← add_path(G_Sol, path_1)
    G_Sol ← add_path(G_Sol, path_2)
    component_nodes[comp] ← add_nodes(component_nodes[comp], path_1)
    component_nodes[comp] ← add_nodes(component_nodes[comp], path_2)
    not_assigned ← subtract_nodes(not_assigned, path_1)
    not_assigned ← subtract_nodes(not_assigned, path_2)
until not_assigned = φ
return G_Sol
```

4.2 Local Search Phase

Once we build a feasible solution to the CmTNSSP, this solution must be improved to approach the global optimal solution. To do this we use a combination of classical local searches, and others based on exact integer linear programming models. There are different strategies for combining a process of building a feasible solution and a set of local searches. In this paper for deploying local searches, we will use a variant of VNS (Variable Neighborhood Search) called VND (Variable Neighborhood Descendant), whose generic algorithm is detailed in (Mladenovic and Hansen, 1997). We have designed five neighborhoods corresponding to the five local searches that we develop below. These local searches are referred to as Extract Insert Nodes **(Extract-Insert)**, Swapping Nodes **(Swapping)**, Components Crossing **(Crossing)**, Best Path with Rays **(Best PWR)** and Best 2-Node-Connected Component **(Best 2NC)** which are applied successively in this order.

4.2.1 Extract-Insert Nodes

This local search performs the extraction of all terminal nodes in a random order from their current positions in the solution, and relocate them to another positions (either in the same component or other) to improve the overall cost without losing feasibility. The extraction procedure is simple, we extract a terminal node and we reconnect the adjacents to the extracted node. To make the insertion of the extracted node we consider the following definition:
Let $i \in T$ a node extracted with T the set of terminal nodes of the graph and a neighborhood N defined as follows:

$$N(i) = \left\{ j \in T : j \quad \begin{array}{l} \text{are the } k \text{ nodes } closer \text{ to node } i \text{ taking into account routing} \\ \text{costs } c_{ij} \text{ defined in original graph } G \end{array} \right\} \quad (29)$$

The loop for each terminal node i, ends after having considered all possible insertions between k closest nodes, and selects the movement that produces the lowest total cost. The algorithm repeats the same procedure for all $i \in T$ not even considered, by examining $N(i)$ until finally selecting the movement that produces the lowest total cost.

4.2.2 Swapping Nodes

This local search selects two nodes and makes an exchange (swapping) between them. This process starts with a random selection of a terminal not pendant node and tests all possible ways to swap this node with another $close$ node belonging to a 2-node-connected component (the same or other). To clarify the concept $close$ we define a neighborhood related to the considered node.

Again we will appeal to the same definition of neighborhood we use in extract-insert local search, (detailed in 4.2.1), i.e. the neighborhood N of k nodes $j \in T$ closest to the node i.
The algorithm begins by taking a random node i and proceeds as follows.
Consider again node j as the nearest node to i. If j is a pendant node, do not perform any movement and continue with the next node, i.e. take a next j closest to i. Each time a swapping movement leads to improvement and keeps the feasibility, the current solution is updated, the possible swapping with other nodes j in descending order of distance are discarded and finally the algorithm continues with the next non pendant terminal node i.

This local search (Algorithm 3) takes two $close$ nodes (as defined in Section 4.2.2), each one in different component, eliminates one of their adjacent edges (for each node) and connects each pair of nodes (in different component) by the edge that generates the best cost.

4.2.3 Best path with pendants

This local search is based on an integer linear programming model. Previously we give a definition of structures used for this local search, that we will call **path with pendant nodes** or shortly **path with pendants**.

Definition 6. ***Path with pendant nodes***. *Given an undirected graph $G = (V, E)$ we say this is a path with pendant nodes with endpoints a and $z \in V$ when exists a path $p(a, z) \subseteq G$ that connects nodes a and z (that we call main path), and the following conditions are met:*

- *G is acyclic and connected.*

- *All nodes that do not belong to p are connected to some node of p through a simple edge.*

Algorithm 3 In this algorithm a terminal node, and one of its closest in another component are selected, adjacent edges of each node are eliminated, and such components are crossed by adding two new edges.

input G_{inic}, T, k
$G_{best} \leftarrow G_{inic}$
for $(i = 1$ **to** $|T|)$ **do**
  if (i is not a pendant node) **then**
    Let K be the ordered set of k nodes closest to node i
    for $(u = 1$ **to** $k)$ **do**
      Let $j = u^{th}$ node closest to node i
      remove an edge adjacent to node i
      remove an edge adjacent to node j
      Let i' be the opposite end of the edge incident to i
      Let j' be the opposite end of the edge incident to a j
      state_1=generate edges (i, j') and (i', j)
      state_2=generate edges (i, j) and (i', j')
      select the state that generates feasible solution with improved resulting cost
      $improve = update(G_{best})$
      if $(improve)$ **then**
        breakfor
        {exit FOR loop, we do not consider next closer nodes}
      end if
    end for
  end if
end for
return G_{best}

Given a feasible solution to the CmTNSSP we should identify all simple cycles that exist in each component and we should explod them in paths, adding their pendants nodes. For each path with pendants, exact local search is applied to obtain the best solution with such topology. This algorithm is based on an integer linear programming model, takes an input graph with two distinguished nodes a and z and returns the best path with pendants with the same endpoints a and z as optimal solution.

We consider the following definitions:

Let $G = (V, E)$ a graph where V is the set of vertices and E is the set of edges.

Let $\hat{T}$ the set of terminal nodes of G.

Let a and z two distinguished terminal nodes such that $a \in \hat{T}$ and $z \in \hat{T}$.

Let $T = \hat{T} \setminus (\{a\} \cup \{z\})$ the set of terminal nodes without a and z.

We define $C = \{c_{ij}\}_{i,j \in V}$ as the routing costs matrix of the graph, for each edge $(i, j$ which belongs to the main path $p(a, z)$.

Let us now define $D = \{d_{ij}\}_{i,j \in V}$ as the connection costs matrix of the graph, that is the cost of the edge (i, j) when one end node is a node of the main path and the other one does not belong to sucha path.

Let $W = V \setminus \hat{T}$ be the set of Steiner nodes. Let us now define the model variables.

$$X_i = \begin{cases} 1 & \text{if node } i \in \hat{T} \text{ belongs to main path} \\ 0 & \text{otherwise} \end{cases}$$

$$Y_i = \begin{cases} 1 & \text{if node } i \in T \text{ is a pendant node} \\ 0 & \text{otherwise} \end{cases}$$

$$Yc_{i,j} = \begin{cases} 1 & \text{if } i \in \hat{T} \text{ and } j \in V \text{ are connected, being } i \text{ a pendant node and } j \text{ a main path node} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{i,j} = \begin{cases} 1 & \text{if edge } (i, j) \text{ is used in the solution} \\ 0 & \text{otherwise} \end{cases}$$

$$w_{i,j} = \begin{cases} 1 & \text{if edge } (i, j) \text{ is a pendant edge and is used in the solution} \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{(i,j)}^{(u,v)} = \begin{cases} 1 & \text{if edge } (i, j) \text{ is used in path that goes from node } u \text{ to node } v \\ 0 & \text{otherwise} \end{cases}$$

The integer linear programming model is defined as follows:

$$min(\sum_{i,j \in V} c_{ij}(x_{ij} - w_{ij}) + \sum_{i,j \in V} d_{ij} w_{ij}) \tag{30}$$

subject to:

$$X_i + Y_i = 1 \qquad \forall i \in T \tag{31}$$

$$X_i = 1 \qquad \forall i \in (\{a\} \cup \{z\}) \tag{32}$$

$$Yc_{ij} \leq X_j \qquad \forall i \in T \quad \forall j \in Adj(i) \tag{33}$$

$$Y_i = \sum_{j \in Adj(i)} Yc_{ij} \qquad \forall i \in T \tag{34}$$

$$\sum_{j \in V} w_{i,j} \leq Y_i \qquad \forall i \in T \tag{35}$$

$$Yc_{i,j} = w_{i,j} \qquad \forall i \in T \quad j \in Adj(i) \tag{36}$$

$$\sum_{j \in Adj(i)} x_{i,j} \leq M(1 - Y_i) + 1 \quad \forall i \in T \quad M \in \mathbb{Z}^{+}, M \geq max(\delta_i) \;\; i = 1 \cdots |V| \tag{37}$$

$$w_{i,j} \leq x_{i,j} \qquad \forall i \in T \quad j \in Adj(i) \tag{38}$$

$$\sum_{j \in Ady[u]} y^{(u,v)}_{(u,j)} = 1 \qquad \forall u, v \in \hat{T}, u \neq v, \tag{39}$$

$$\sum_{i \in Ady[v]} y^{(u,v)}_{(i,v)} = 1 \qquad \forall u, v \in \hat{T}, v \neq u, \tag{40}$$

$$\sum_{i \in Ady[p]} y^{(u,v)}_{(i,p)} - \sum_{i \in Ady[p]} y^{(u,v)}_{(p,i)} \geq 0 \quad \forall u, v \in \hat{T}, \quad \forall p \in V \setminus u, v \tag{41}$$

$$y^{(u,v)}_{(i,j)} + y^{(u,v)}_{(j,i)} \leq x_{i,j} \quad \forall u, v \in \hat{T}, u \neq v, \quad \forall (i,j) \in E \tag{42}$$

$$Y_i = 0 \qquad \forall i \in W \tag{43}$$

$$\sum_{j \in Ady(i)} Yc_{i,j} = 0 \qquad \forall i \in W \tag{44}$$

$$(\sum_{i \in Ady[j]} Yc_{i,j} + 2X_j - \sum_{i \in Ady[j]} x_{j,i} = 0) \qquad \forall j \in W \tag{45}$$

$$\sum_{i \in Ady[j]} (Yc_{i,j} + Yc_{j,i}) + 2X_j - \sum_{i \in Ady[j]} x_{j,i} = 0 \qquad \forall j \in T \tag{46}$$

$$\sum_{i \in Ady[j]} (Yc_{i,j}) + X_j - \sum_{i \in Ady[j]} x_{j,i} = 0 \qquad \forall j \in (\{a\} \cup \{z\}) \tag{47}$$

Algorithm 4 describes local search which involves the replacement of a path with pendants, by other with the same nodes and endpoints whose total cost is lower (optimal). It begins by taking as input the graph G_{Sol}, feasible solution of CmTNSSP. For each m components of G_{Sol} we count its cycles, which are them identified and stored in the indexed list all_cycles (Lines 3 and

Algorithm 4 Improved solution decomposing all cycles of the graph on paths with their respective pendant nodes, getting the best substitute for each of them.

1: input $G_{sol},\ G,\ C,\ D,\ T,\ MAX\_PATH\_LENGTH$
2: $G_{best} \leftarrow G_{sol}$
3: $q\_cycles = cycles\_count(G_{sol})$ {Numbers of cycles of G_{sol}}
4: $all\_cycles \leftarrow cycles(G_{sol})$ {Array with cycles of G_{sol}}
5: **for** ($i = 1$ **to** q_cycles) **do**
6: $\quad path\_long$ = min (length($all\_cycles(i)$), MAX_PATH_LENGTH))
7: $\quad begin\_path = 1$
8: $\quad end\_path$ =length($all\_cycles(i)$)
9: $\quad$ **while** ($end\_path \leq$ length($all\_cycles(i)$)) **do**
10: $\qquad end\_path$ = $begin\_path$ + 3 + (rand() MOD ($path\_long$ - 2))
11: $\qquad P$ = **path_with_rays**($G_{sol},\ all\_cycles(i),\ begin\_path,\ (end\_path$ MOD length($all\_cycles(i)$)
12: $\qquad H \leftarrow$ **induced_graph_path**($P,\ G,$ T)
13: $\qquad P_{best} =$ **best_pwr**($G_{sol},\ G\ P\ C,\ D,\ H$)
14: $\qquad G_{best} \leftarrow G_{best}$ - P + P_{best}
15: $\qquad begin\_path = end\_path$
16: $\quad$ **end while**
17: **end for**
18: **return** G_{best}

4). Next, each of the cycles identified in the previous steps are treated, running the operations defined in the scope of **for** (Lines 5 to 17) until exhaust all cycles. Each cycle is divided into a certain number of paths of variable length (MAX_PATH_LENGTH parameter). We set a start node and an end node of the first path in the cycle (Lines 7 and 8).

Once initialized the path to process, we enter into a repetitive loop determined by scope of (**while**) (Lines 9 to 16) which readjust path length in a random way (Line 10).

Each path obtained in the previous step is added with pendant nodes present in G_{Sol} (Line 11) obtaining a path with endpoints $begin\_path$, end_path and pendant nodes, such we specify in Definition 6.

In the next step we generate the graph H induced by nodes of the path with pendants P respect to the original graph G. (Line 12). Graph H thus generated is input of process **best_pwr** that gives us the best path with pendants with ends $begin\_path$, end_path (Line 13).

In line 14 we perform the substitution of the path with pendants P by the path with pendants P_{best} obtaining a better solution cost G_{best}. Next, we reset the start and the end node in the cycle we are processing, (Line 15) to generate a new path. After processing all paths within each cycle, we return the best cost solution G_{best} (Line 18).

4.2.4 Best 2-Connected Component

This local search is also based on integer linear programming. Just as in the previous local search, given a feasible solution to the problem, Algorithm 5 identifies all cycles that exist in each component. For each cycle we will now apply an exact algorithm getting the best replacement solution that changes a cycle by 2-node-connected topology.

As we saw in Section 1, the best 2-node-connected solution covering a certain set of nodes is not necessarily a cycle, so this local search may include such topologies in our solution (see Figure 1). This algorithm takes as input the induced sub-graph of the original graph with nodes of the cycle and some Steiner nodes, and returns the best 2-connected sub-graph, i.e it can potentially changes a cycle for a structure that contains a Monma's graph if such change improves solution costs.

We use to model this local search a particular case of **GSP** (General Steiner Problem) where connectivity of all its terminal nodes is two. We consider th following definitions:

Let $G = (V, E)$ and undirected graph where V is the set of vertices and E the set of edges of graph G.

Let $\hat{T}$ set of terminal nodes of graph G.

Define $C = \{c_{ij}\}_{i,j \in V}$ as the routing cost matrix, i.e. costs when edge (i, j) belongs the two-node-connected structure of the component. Only use this cost matrix as in this local search are not considered pending nodes that have been generated so far.

Let us define below the model variables.

$$x_{i,j} = \begin{cases} 1 & \text{if edge } (i,j) \text{ is used in the solution} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{(i,j)}^{(u,v)} = \begin{cases} 1 & \text{if edge } (i,j) \text{ is used in a path from node } u \text{ to } v \\ 0 & \text{otherwise} \end{cases}$$

Once the variables are specified, the integer linear programming model is defined as follows:

$$min(\sum_{i,j \in V} c_{ij} x_{ij})$$

subject to:

$$\sum_{j \in Ady[u]} y_{(u,j)}^{(u,v)} = 2 \qquad \forall u, v \in \hat{T}, u \neq v,$$

$$\sum_{i \in Ady[v]} y_{(i,v)}^{(u,v)} = 2 \qquad \forall u, v \in \hat{T}, v \neq u,$$

$$\sum_{i \in Ady[p]} y_{i,p}^{(u,v)} - \sum_{i \in Ady[p]} y_{(p,i)}^{(u,v)} \geq 0 \quad \forall u, v \in \hat{T}, \quad \forall p \in V \setminus u, v$$

$$y_{(i,j)}^{(u,v)} + y_{(j,i)}^{(u,v)} \leq x_{i,j} \quad \forall u, v \in \hat{T}, u \neq v, \quad \forall (i,j) \in E$$

Algorithm 5 Improving solution changing cycles of the graph for the best 2-node-connected component.

1: input $G,\ G_{sol},\ C,\ T$
2: $G_{best} \leftarrow G_{sol}$
3: $q\_cycles = cycles\_count(G_{sol})$ {Number of cycles of G_{sol}}
4: $all\_cycles \leftarrow cycles(G_{sol})$ {Array with cycles of G_{sol}}
5: **for** $(i = 1$ **to** $q\_cycles)$ **do**
6: $\quad best =$ **best_2nc**$(G_{sol},\ G_{orig},\ all\_cycles(i))$
7: $\quad G_{best} \leftarrow G_{best}$ - $all\_cycles(i)$ + $best\_2nc$
8: **end for**
9: **return** G_{best}

Analogous to Algorithm 4, Algorithm 5 counts and identifies the cycles present in G_{sol} (lines 3 and 4). For each of these cycles the process **best_2nc** (line 6) returns the best 2-node-connected structure and in the line 7 performs substitution of cycle by the best one.

5 Computational results

To the best of our knowledge, it does not exist an exact resolution of the problem CmTNSSP in the literature, therefore in principle we do not have a reference to compare the effectiveness of the metaheuristic developed in this work. Considering that the CmTNSSP is a relaxation of CmRSP and that any solution of CmRSP is also solution of CmTNSSP, we refer to the work on the CmRSP by Baldacci et al. (2007). In that paper, the vast majority of the problem instances used are solved to optimality and those that are unresolved have defined lower bounds that will guide us to measure the results generated by our application. Also, we compare against more recent results for CmRSP provided by Naji-Azimi et al. (2012).

The exact ILP model was coded in AMPL and ran in CPLEX 12.7. The heuristic was coded in C, using the callable library of CPLEX. Hardware where algorithms were run, consists of a computer with Intel I7 processor with 8 Gb. RAM and OS Fedora Core 20.

5.1 Exact resolution

The model has been implemented and executed on several small instances and we have selected one to describe, e.g. we have defined a graph called $nut30$ and denoted $N = (V, E)$ where $V = T \cup W \cup \{d\}$, where: $T = \{1 \cdots 19\}$ the set of terminal nodes of graph N, $W = \{20 \cdots 29\}$ the set of Steiner nodes and $d = \{0\}$ the depot node, capacity constraint $Q = 2$ and the number of components $m = 2$. Routing cost matrix C and connection costs matrix D are identicals and values are the euclidean distances between vertices of the graph N.

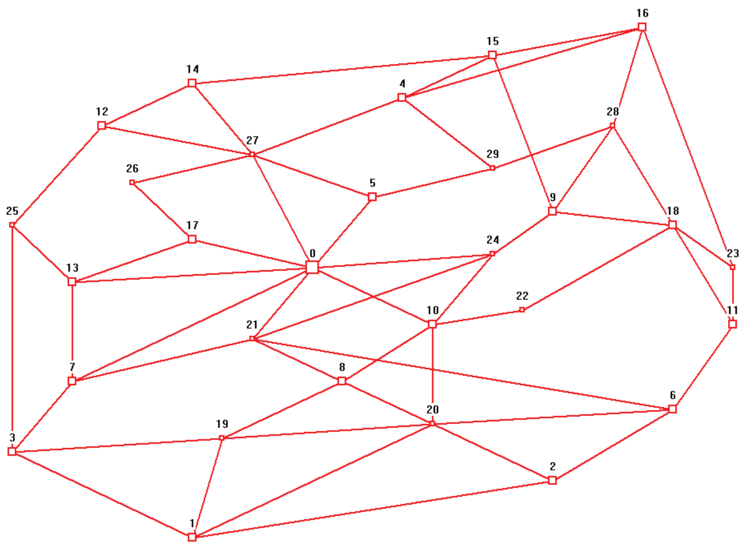

Figure 4: Initial graph ($nut30$) for testing ILP model of CmTNSSP.

In order to shorten the computational processing used in executing the solver CPLEX, we have not considered the complete graph and we have generated only some edges of the graph N, hence the set E contains only the edges that can be seen in Figure 4. Still, given the complexity of the model, the transformation to an integer linear programming for this instance had 721,244 rows, 618,913 columns and 629,149 non-zero values. After running the model we obtain the exact solution of CmTNSSP for the instance defined above. We can observe the graphical representation in Figure 5. Note that even though we are solving the CmTNSSP, the optimal solution is also a solution of CmRSP, i.e. the connected components are exclusively cycles.

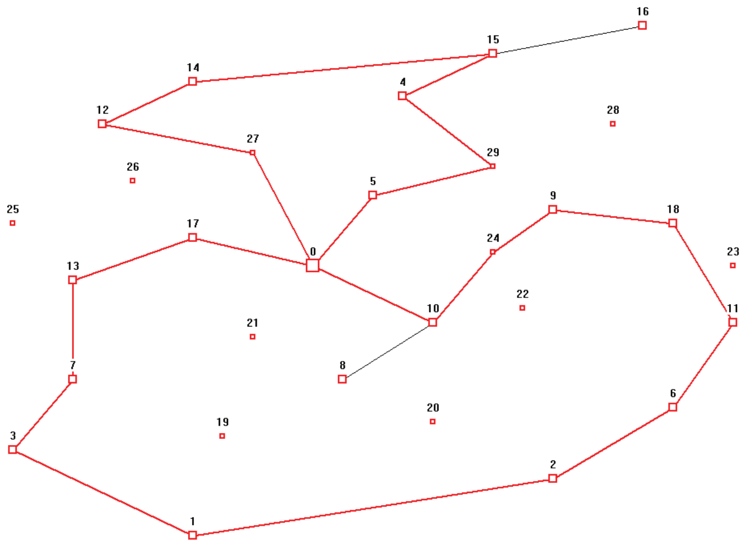

Figure 5: Global optimum of CmTNSSP for $nut30$, found using CPLEX solver.

5.1.1 Resolution by GRASP

We use the test instances proposed by Baldacci et al. (2007), which are divided into two classes, A and B. In class A routing costs and connection costs match. In class B routing costs are greater than connection costs. For both classes of instances graphs used are the same, the only difference is in the cost of its edges according to whether or not incident to a pendant node. These graphs are $eil51$, $eil76$ and $eil101$ obtained from the TSPLIB, the Traveling Salesman Problem Library Reinelt (1990). Additionally, a new graph called $eil26$ is added and it is built with the first 26 vertices of $eil51$. Then we set $n = \{26, 51, 76, 101\}$ as the total of vertices for each of the graphs defined in the previous paragraph. First node of each of these graphs is tagged as depot. The remaining 25, 50, 75 and 100 respectively are divided into terminal and optional nodes according to a parameter $\alpha \in \{0.25, 0.5, 0.75, 1\}$, where U (set of terminal nodes) are the first $\alpha(n-1)$ nodes and W (set of Steiner nodes) the remaining. For each of these combinations we will generate instances with $m \in \{3, 4, 5\}$ and Q will be calculated for a use of the components above the 90 % using the following formula:

$$Q = \left\lceil \frac{|U|}{0.9m} \right\rceil \tag{48}$$

With regard to costs for classes of instances A and B these are defined in the following way:

- **Class A**. Routing and connection costs are equal and correspond to the Euclidean distance $e_{i,j}$ between nodes (i,j) considered. Thus $c_{i,j} = d_{i,j} = e_{i,j}$
- **Class B**. Routing costs $c_{i,j}$ = $\lceil \beta\, e_{i,j} \rceil$, where β is an integer in the range [6,9]. Connection costs will be $d_{i,j}$ = $\lceil (10-\beta)e_{i,j} \rceil$. For our Class B instances we use $\beta = 7$.

In addition to the definitions specified in the preceding paragraphs, there is another constraint on connection costs. Each edge connecting node on a 2-node-connected component with a pendant node cannot have a higher cost to a certain bound:

$$d_{max} = 0.2 \times \frac{\sum_{(i,j)\in E} d_{ij}}{|E|} \tag{49}$$

This is in fact an additional problem constraint, which is also present in the studies used as reference for comparison in this work.

We can see in Table 1 the results of the solutions for Class A instances. The notations corresponding to each column are the following:

$|T|$ is the number of terminal nodes in the specified instance, CN is the number of nodes present in 2-node-conected structures of components, PN is the number of pendant nodes in the solution, SN is the number of Steiner nodes used in the solution, Z_{best} is the optimum value found by GRASP, $\bar{Z}_1$ is the reference optimum value obtained in Baldacci et al. (2007), $\bar{Z}_2$ is the best value obtained in recent work by Naji-Azimi et al. (2012) and gap is the percentage difference of $\bar{Z}_1$ with our calculated solution, and it is calculated as follows:

$$gap = \frac{Z_{best} - \bar{Z}_1}{\bar{Z}}$$

INSTANCE	$\|T\|$	Q	*CN*	*PN*	*SN*	Z_{best}	$\bar{Z}_1$	$\bar{Z}_2$	*gap %*	*t(s)*
A01-n026-m03	12	5	12	0	1	242	242	242	0,000	1.61
A02-n026-m04	12	4	12	0	1	261	261	261	0,000	0.97
A03-n026-m05	12	3	12	0	1	292	292	292	0,000	13.77
A03-n026-m05	12	3	12	0	0	292	292	292	0,000	4.54
A04-n026-m03	18	7	18	0	0	301	301	301	0,000	34.29
A05-n026-m04	18	5	18	0	0	339	339	339	0,000	62.58
A05-n026-m04	18	5	18	0	1	339	339	339	0,000	9.34
A06-n026-m05	18	4	18	0	0	375	375	375	0,000	2.67
A07-n026-m03	25	10	24	1	0	325	325	325	0,000	14.06
A08-n026-m04	25	7	25	0	0	362	362	362	0,000	3.99
A10-n051-m03	12	5	12	0	0	242	242	242	0,000	20.09
A11-n051-m04	12	4	12	0	3	261	261	261	0,000	6.42
A12-n051-m05	12	3	11	1	2	286	286	286	0,000	37.69
A13-n051-m03	25	10	22	3	3	322	322	322	0,000	130.85
A14-n051-m04	25	7	24	1	1	360	360	360	0,000	49.75
A15-n051-m05	25	6	23	2	2	379	379	379	0,000	117.67
A16-n051-m03	37	14	33	4	1	373	373	373	0,000	296.60
A17-n051-m04	37	11	33	4	1	405	405	405	0,000	80.49
A18-n051-m05	37	9	33	4	1	432	432	432	0,000	2720.60
A19-n051-m03	50	19	45	5	0	458	458	458	0,000	1674.86
A20-n051-m04	50	14	48	2	0	490	490	490	0,000	3429.11
A21-n051-m05	50	12	43	7	0	520	520	520	0,000	6338.64
A22-n076-m03	18	7	17	1	5	330	330	330	0,000	36.13
A23-n076-m04	18	5	15	3	7	385	385	385	0,000	112.97
A24-n076-m05	18	4	17	1	4	448	448	448	0,000	109.91
A25-n076-m03	37	14	35	2	2	403	402	402	0,249	3624.35
A26-n076-m04	**37**	**11**	**36**	**1**	**3**	**456**	**460**	**457**	**-0,870**	**7200.00**
A27-n076-m05	37	9	36	1	4	483	479	479	0,835	7200.00
A28-n076-m03	56	21	48	8	1	474	471	471	0,637	7200.00
A29-n076-m04	**56**	**16**	**49**	**7**	**1**	**519**	**523**	**519**	**-0,765**	**7200,00**
A30-n076-m05	56	13	50	6	2	547	545	545	0,367	7200.00
A31-n076-m03	75	28	71	4	0	571	564	564	1,241	7200.00
A32-n076-m04	75	21	73	2	0	617	606	602	1,815	7200.00
A33-n076-m05	**75**	**17**	**68**	**7**	**0**	**651**	**654**	**640**	**-0,459**	**7200.00**
A34-n101-m03	25	10	21	4	7	363	363	363	0,000	199.27
A35-n101-m04	25	7	21	4	9	415	415	415	0,000	1023.84
A36-n101-m05	25	6	22	3	9	448	448	448	0,000	1264.62
A37-n101-m03	50	19	46	4	8	500	500	500	0,000	4020.65
A38-n101-m04	50	14	47	3	6	538	532	528	1,128	7200.00
A39-n101-m05	50	12	46	4	5	573	568	567	0,880	7200.00
A40-n101-m03	75	28	69	6	5	613	595	595	3,025	7200.00
A41-n101-m04	75	21	73	2	1	651	625	623	4,160	7200.00
A42-n101-m04	75	17	70	5	2	677	662	657	2,266	7200.00
A43-n101-m03	100	38	84	16	0	662	646	646	2,477	7200.00
A44-n101-m04	100	28	87	13	0	680	680	679	0,000	7200.00
A45-n101-m05	100	23	84	16	0	713	700	700	1,857	7200.00

Table 1: Best values found for instances Class A.

Finally column $t(s)$ points the maximum processing time of the instance in seconds. We have defined a limit of 7200 seconds of maximum runtime.

Such Table 1 reports the best Z_{best} found for CmTNSSP. Values in bold are those where the proposed GRASP based heuristic improves the solution found by the original work of Baldacci et al. (2007). Note that some of those values where later improved by Naji-Azimi et al. (2012). In general terms, we can conclude that our proposed algorithm is successful in solving the CmRSP, a problem closely related to CmTNSSP. Also, some improvements in specific instances were found.

Similarly in Table 2 we can see the best values generated by our algorithm for Class B instances. We can observe even more improvements with respect to the original work of Baldacci et al. (2007) and similar relationship with results of Naji-Azimi et al. (2012). The same conclusions already stated for Class A, also hold for Class B instances. Other results about this work and more detailed procedures with other instances can be read in Bayá (2014).

It is worth mentioning that, due to lack of references for comparison, we are comparing against results produced by algorithms which were not conceived to solve the problem introduced in this work. Nevertheless, our results are competitive when compared with the ones produced by the authors who introduced the CmRSP. The comparison against more recent results gives less chances to succeed in terms of improvements on CmRSP instances, since newer heuristic solving methods are very much specialized. Actually, the best known results for the CmRSP have been published very recently by (Zhang, Qin and Lim, 2014), a work which is contemporary with this one.

INSTANCE	$\lvert T \rvert$	Q	*CN*	*PN*	*SN*	Z_{best}	$\bar{Z}_1$	$\bar{Z}_2$	*gap %*	*t(s)*
B01-n026-m03	12	5	11	1	1	1684	1684	1684	0,000	3.09
B02-n026-m04	12	4	12	0	1	1827	1827	1827	0,000	1.09
B03-n026-m05	12	3	11	1	2	2041	2041	2041	0,000	10.68
B04-n026-m03	18	7	17	1	1	2104	2104	2104	0,000	24.90
B05-n026-m04	18	5	17	1	1	2370	2370	2370	0,000	78.21
B06-n026-m05	18	4	17	1	2	2615	2615	2615	0,000	47.01
B07-n026-m03	25	10	24	1	0	2251	2251	2251	0,000	35.13
B08-n026-m04	25	7	24	1	0	2510	2510	2510	0,000	51.65
B09-n026-m05	25	6	25	0	0	2674	2674	2674	0,000	150.31
B10-n051-m03	12	5	10	2	2	1681	1681	1681	0,000	2035.19
B11-n051-m04	12	4	10	2	3	1821	1821	1821	0,000	49.26
B12-n051-m05	12	3	10	2	2	1975	1972	1972	0,152	930.42
B13-n051-m03	25	10	21	4	3	2176	2176	2176	0,000	1724.28
B14-n051-m04	25	7	22	3	3	2470	2470	2470	0,000	626.97
B15-n051-m05	25	6	21	4	4	2579	2579	2579	0,000	92.66
B16-n051-m03	37	14	29	8	2	2490	2490	2490	0,000	3699.45
B17-n051-m04	37	11	29	8	2	2735	2721	2721	0,515	3605.47
B18-n051-m05	37	9	32	5	2	2908	2908	2908	0,000	197.51
B19-n051-m03	50	19	39	11	0	3015	3015	3015	0,000	871.33
B20-n051-m04	50	14	39	11	0	3267	3260	3260	0,215	7200,00
B21-n051-m05	50	12	38	12	0	3404	3404	3404	0,000	3773.22
B22-n076-m03	18	7	15	3	4	2253	2253	2253	0,000	186.10
B23-n076-m04	18	5	13	5	8	2620	2620	2620	0,000	90.78
B24-n076-m05	18	4	15	3	9	3155	3059	3059	3,138	7200,00
B25-n076-m03	37	14	32	5	6	2731	2720	2720	0,404	7200,00
B26-n076-m04	**37**	**11**	**34**	**3**	**4**	**3134**	**3138**	**3100**	**-0,127**	**7200.00**
B27-n076-m05	37	9	36	1	3	3329	3311	3284	0,544	7217.19
B28-n076-m03	**56**	**21**	**40**	**16**	**4**	**3044**	**3088**	**3044**	**-1,425**	**7200.00**
B29-n076-m04	**56**	**16**	**44**	**12**	**2**	**3439**	**3447**	**3415**	**-0,232**	**7200.00**
B30-n076-m05	**56**	**13**	**44**	**12**	**2**	**3635**	**3648**	**3632**	**-0,356**	**3797.03**
B31-n076-m03	**75**	**28**	**55**	**20**	**0**	**3724**	**3740**	**3652**	**-0,428**	**2112.23**
B32-n076-m04	75	21	57	18	0	4096	4026	3964	1,739	7200,00
B33-n076-m05	75	17	58	17	0	4489	4288	4217	4,688	7200,00
B34-n101-m04	25	7	19	6	9	2445	2434	2434	0,369	7200,00
B35-n101-m04	25	7	19	6	6	2795	2782	2782	0,467	7200,00
B36-n101-m05	25	6	18	7	4	3009	3009	3009	0,000	597.71
B37-n101-m03	**50**	**19**	**40**	**10**	**8**	**3331**	**3332**	**3322**	**-0,030**	**7200,00**
B38-n101-m04	50	14	38	12	8	3560	3533	3533	0,764	7200,00
B39-n101-m05	50	12	41	9	8	3873	3872	3834	0,026	7200,00
B40-n101-m03	75	28	68	7	5	3931	3923	3887	0,204	7200,00
B41-n101-m04	75	21	68	7	6	4332	4125	4082	5,018	7200,00
B42-n101-m05	75	17	69	6	6	4494	4458	4358	0,808	7200,00
B43-n101-m03	100	38	96	4	0	4403	4110	4110	7,129	7200,00
B44-n101-m04	100	28	95	5	0	4526	4506	4355	0,444	7200,00
B45-n101-m05	100	23	96	4	0	4639	4632	4565	0,151	7200,00

Table 2: Best values found for instances Class B

5.2 CmTNSSP with non cyclical 2-node-connected components

In results displayed in Tables 1 and 2, despite local search applied inducing the use of non-cyclical 2-node-connected components if these are optimal (see Section 4.2.4), we didn't find such structures for tested instances. To verify that the proposed algorithm finds such solutions we generate an additional test case. We are given graph $G = (V, E)$ complete, with $|V| = 36$ and nodes distributed in the following way:

$$d = \{0\}, \quad T = \{1 \cdots 27\}, \quad W = \{28 \cdots 35\}$$

The set of vertices V are located on a planar coordinate system (x, y) with the following values:

0 (11,9)	6 (5,9)	12 (14,12)	18 (20,6)	24 (25,9)	30 (3,10)
1 (9,13)	7 (3,7)	13 (14,6)	19 (21,17)	25 (28,12)	31 (16,5)
2 (7,11)	8 (8,8)	14 (16,9)	20 (21,12)	26 (28,6)	32 (21,10)
3 (6,13)	9 (7,6)	15 (18,12)	21 (22,9)	27 (30,9)	33 (22,14)
4 (3,12)	10 (4,4)	16 (19,9)	22 (24,6)	28 (7,9)	34 (25,5)
5 (1,9)	11 (13,15)	17 (17,6)	23 (25,12)	29 (9,4)	35 (28,9)

Cost matrices $C = \{c_{ij}\}_{i,j \in V}$ y $D = \{d_{ij}\}_{i,j \in V}$ are both defined by Euclidian distances between vertices i, j multiplied by a factor 10, except for a set of edges $E' \subseteq E$ to which is assigned the following costs:

$c_{0,11} = c_{11,0} = d_{0,11} = d_{11,0} = 1$

$c_{12,15} = c_{15,12} = d_{12,15} = d_{15,12} = 5$

$c_{0,14} = c_{14,0} = d_{0,14} = d_{14,0} = 1$

$c_{16,14} = c_{14,16} = d_{16,14} = d_{14,16} = 1$

$c_{0,13} = c_{13,0} = d_{0,12} = d_{12,0} = 1$

$c_{0,13} = c_{13,0} = d_{0,13} = d_{13,0} = 1$

$c_{0=15,20} = c_{20,15} = d_{15,20} = d_{20,15} = 1$

$c_{22,26} = c_{26,22} = d_{22,26} = d_{26,22} = 1$

$c_{20,23} = c_{23,20} = d_{20,23} = d_{23,20} = 1$

$c_{18,22} = c_{22,18} = d_{18,22} = d_{22,18} = 1$

$c_{14,17} = c_{17,14} = d_{14,17} = d_{17,14} = 80$

$c_{18,17} = c_{17,18} = d_{18,17} = d_{17,18} = 1$

$c_{24,27} = c_{27,24} = d_{24,27} = d_{27,24} = 5$

Constructor parameters are the following:

$m = 2; \quad Q = 18; \quad ListSize = 4$

$$k = 7; \quad p = 11; \quad MAX\_PATH\_LENGTH = 4$$

For values specified above, GRASP-VND algorithm found optimal (local to our knowledge) feasible solution with a non-cyclical structure in one of its components (Figure 6)

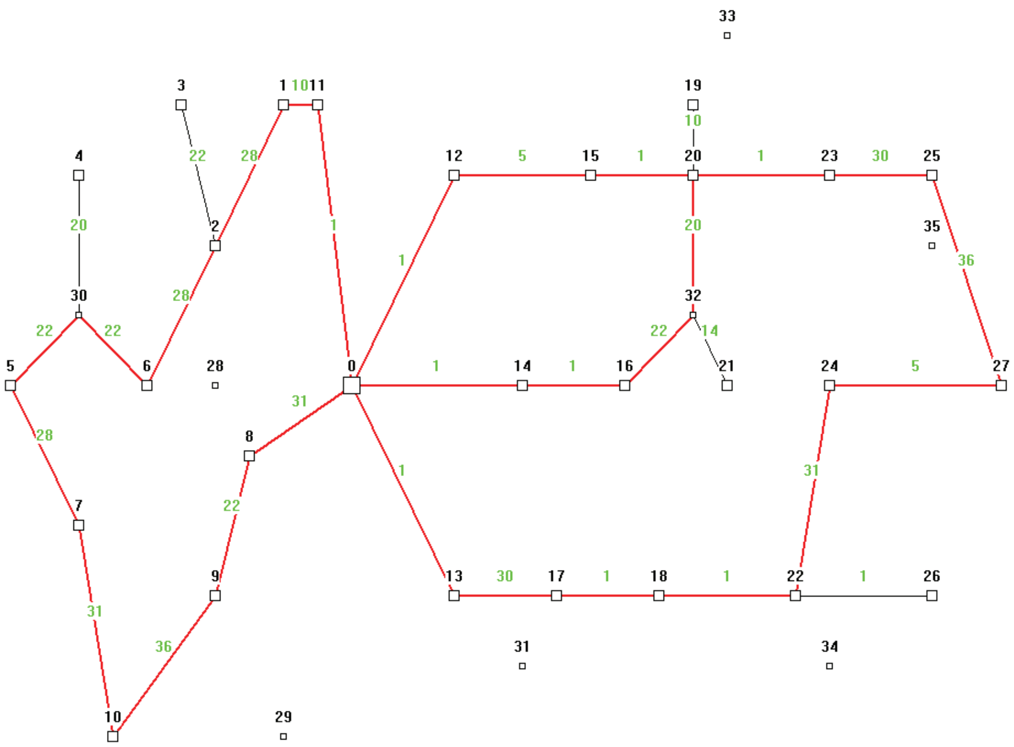

Figure 6: Topology of non-cyclical 2-node-connected component found.

These results show that designed GRASP-VND metaheuristic is able to obtain the best solution (local optimum) with non-cyclical structures.

6 Conclusions and future works

The Capacitated m Two-Node Survivable Star Problem (CmTNSSP) has been introduced. As far as we know, it has not been studied in prior literature. The need for redundancy and cheaper costs in network deployment is remarkable. Inspired in predictions from Clyde Monma and the previous CmRSP, we propose an alternative problem, where rings are replaced by arbitrary two-node connected components. Both problems are computationally intractable. Therefore, heuristics are suitable for large case scenarios. The CmTNSSP has been modeled by an ILP formulation and heuristically addressed following a GRASP metaheuristic enriched with a variable neighborhood descend (VND) and exact local searches. Numerical results validated both the exact formulation and the heuristic approach. Results from the literature concerning CmRSP were taken as reference for comparison. In all cases, the components obtained were cycles instead of other two-connected topologies. We found that a particular cost structure lead to non-cyclical solutions. Further research is needed in order to understand the nature of problem instances which influence these results.

In this paper we have seen the CmTNSSP as a slight variation of CmRSP. However delay-sensitive applications can increase the relevance of CmTNSSP with respect to CmRSP. To achieve this goal, diameter constraints should be introduced to ensure connectivity of any pair of nodes by a limited number of hops. Obviously there will be a trade-off when this constraint is added to the problem. Two-node-connected components (not purely cycles) can meet this objetive from a topological point of view. Adding diameter constraints become CmTNSSP in a more sophisticated problem, covering other network requirements such as quality of service (QoS). Authors are actually researching this line of work. As a future work, we also wish to apply these techniques to the design of real-life networks.

References

Baldacci, R., Dell'Amico, M. and González, J. J. S.: 2007, The capacitated m-ring-star problem, *Operations Research* **55**(6), pp. 1147–1162.

Bayá, G.: 2014, *Diseño Topológico de Redes. Caso de Estudio: Capacitated* m *Two-Node Survivable Star Problem*, Master's thesis, Universidad de la República. Pedeciba Informática, Montevideo, Uruguay.

Bhandari, R.: 1997, Optimal physical diversity algorithms and survivable networks, *Computers and Communications, 1997. Proceedings., Second IEEE Symposium on*, pp. 433–441.

Dantzig, G. B. and Fulkerson, R.and Johnson, S. M.: 1954, Solution of a large-scale traveling salesman problem, *Operations Research* **2**(4), pp. 393–410.

Feo, T. A. and Resende, M.: 1995, Greedy randomized adaptive search procedures, *Journal of Global Optimization* **6**(2), 109–133.

Frederickson, G. N. and Ja'Ja', J.: 1981, Approximation algorithms for several graph augmentation problems, *SIAM Journal on Computing* **10**(2), 270–283.

Hoshino, E. A. and De Souza, C. C.: 2012, A branch-and-cut-and-price approach for the capacitated m-ring-star problem, *Discrete Appl. Math.* **160**(18), 2728–2741.

Labbé, M., Laporte, G., Martín, I. R. and González, J. J. S.: 2004, The ring star problem: Polyhedral analysis and exact algorithm, *Networks* **43**(3), 177–189.

Labbé, M., Laporte, G., Martín, I. R. and González, J. J. S.: 2005, Locating median cycles in networks, *European Journal of Operational Research* **160**(2), 457 – 470. Decision Support Systems in the Internet Age.

Mauttone, A., Nesmachnow, S., Olivera, A. and Robledo, F.: 2008, Solving a ring star problem generalization, *Conference: 2008 International Conferences on Computational Intelligence for Modelling, Control and Automation (CIMCA 2008), Intelligent Agents, Web Technologies and Internet Commerce (IAWTIC 2008), Innovation in Software Engineering (ISE 2008).*, pp. 981–986.

Mladenovic, N. and Hansen, P.: 1997, Variable neighborhood search, *Computers & Operations Research* **24**(11), 1097 – 1100.

Monma, C. L., Munson, B. S. and Pulleyblank, W.: 1990, Minimum-weight two-connected spanning networks, *Mathematical Programming* **46**(1-3), 153–171.

Naji-Azimi, Z., Salari, M. and Toth, P.: 2010, A heuristic procedure for the capacitated m-ring-star problem, *European Journal of Operational Research* **207**(3), 1227 – 1234.

Naji-Azimi, Z., Salari, M. and Toth, P.: 2012, An integer linear programming based heuristic for the capacitated m-ring-star problem, *European Journal of Operational Research* **217**(1), 17–25.

Reinelt, G.: 1990, TSPLIB - A t.s.p. library, *Technical Report 250*, Universität Augsburg, Institut für Mathematik, Augsburg.

Resende, M. G.: 2009, Greedy randomized adaptive search procedures, *in* C. A. Floudas and P. M. Pardalos (eds), *Encyclopedia of Optimization*, Springer US, pp. 1460–1469.

Richey, M. B.: 1990, Optimal location of a path or tree on a network with cycles, *Networks* **20**(4), 391–407.

Robledo, F.: 2005, *GRASP heuristics for Wide Area Network design*, PhD thesis, INRIA.
URL: *http://www.fing.edu.uy/inco/pedeciba/bibliote/tesis/tesisd-robledo.pdf*

Stoer, M.: 1992, *Design of Survivable Networks (Lecture Notes in Mathematics).*, Springer.

Zhang, Z., Qin, H. and Lim, A.: 2014, A memetic algorithm for the capacitated m-ring-star problem, *Appl. Intell.* **40**(2), 305–321.

Zorpette, G.: 1989, Keeping the phone lines open, *Spectrum, IEEE* **26**(6), 32–36.