UNIVERSITÀ DI SALERNO
Dipartimento di Matematica e Informatica
*D.M.I.*
Via Ponte don Melillo – 84084 Fisciano (SA) – Italy

# Experimental Comparison of Algorithms for Bounded-Degree Spanning Tree Problems

R. Cerulli, M. Gentili, A. Iossa

Preprint n. 23 – 2005

# Experimental Comparison of Algorithms for Bounded-Degree Spanning Tree Problems

R. Cerulli [a], M. Gentili [a,*], A. Iossa [b]

[a] *Dipartimento di Matematica ed Informatica, Università di Salerno, 84081 Baronissi (SA), Italy*
*{raffaele, mgentili }@unisa.it*

[b] *Dipartimento di Statistica, Probabilità e Statistiche Applicate, Università degli studi di Roma "La Sapienza", 00185 Roma (RM), Italy*
*aiossa@unisa.it*

**Abstract**

Given a connected graph $G$ a vertex is said to be *branch* if its degree is greater than 2. We consider two problems arising in the context of optical networks:
(i) finding a spanning tree of $G$ with the minimum number of branch vertices and
(ii) finding a spanning tree of $G$ such that the degree sum of the branch vertices is minimized.
For these NP-hard problems, heuristics, that give good quality solutions, do not exist in the literature. In this paper we analyze the relation between the problems, provide a single commodity flow formulation to solve the problems by means of a solver and develop different heuristic strategies to compute feasible solutions that are compared with the exact ones. Our extensive computational results show the algorithms to be very fast and effective.

*Key words:* Spanning trees, Spanning Spiders, Branch Vertices, Heuristics, Optical Networks

## 1 Introduction

There are several variants of the spanning tree problem that are useful to model problems arising in communication networks. For example, the network may be required to connect a specified subset of nodes (Steiner Tree Problem [5]); if a measure is assigned with each link, one could be interested in looking

* Corresponding author.

for homogeneous subgraphs of the network (Minimum Labelling Spanning Tree Problem [1],[3]); in optical networks it is useful to connect the nodes in a way such that the number of connections of each node is limited (Spanning Tree with Minimum Number of Branch Nodes [2] ). In this paper we focus both on this latter problem and a related problem arising in optical networks.

In particular, in an optical network, the wave division multiplexing technology allows to propagate different light beams on the same optical fiber, as long as they use a different fixed wavelength. Multicast technology on an optical network permits to replicate the optical signal from one source to many destination nodes by means of a network device (*switch*) that permits to replicate a signal, *splitting* light. Many applications, such as world wide web browsing, video conferences etc., require such a technology for efficiency purposes. A *light-tree* connects a node to a subset of nodes in the network, allowing multicasting communications. The nodes of the tree whose degree is greater than 2 are called *branch* nodes. Switches are located on such branch nodes of the tree. Since, optical networks have a limited number of these switches, it is important to determine the minimum number of switches to locate on the network in order to guarantee all the multicast connections. Given a connected graph $G$ representing a network, the minimum number of light-splitting switches can be determined by looking for the spanning tree of $G$ with the minimum number of branch vertices (in the sequel the problem will be referred to as MBV).

Such a problem has been recently addressed in Gargano et al. [2] where the computational complexity of the problem is studied. In particular, let $s(G)$ denote the smallest number of branch vertices in any spanning tree of $G$. A graph $G$ with $s(G) \leq 1$ is said to admit a *spanning spider*. The problem of deciding whether a graph $G$ admits a spanning spider is shown to be NP-complete. However, when the degree sum of any three independent vertices of $G$ is greater than the total number of nodes minus one, i.e., $\delta_3(G) \geq n-1$, then $G$ is proved to admit a spanning spider and a polynomial time algorithm to find it is provided. The more general decisional problem of deciding whether $s(G) \leq k$, for any integer $k$, is also addressed and shown to be NP-complete on general graphs and cubic graphs.

The only existing algorithm to solve MBV is the one given in [2], that exactly computes in polynomial time the solution on a class of graphs satisfying the above mentioned density conditions. Such an algorithm cannot be used (neither extended) to solve the problem on general graphs and, moreover, in applications that arise in real-world situations, it is often the case that the network does not satisfy such density conditions. Therefore, it is of great importance to be able to deal with such real-case situations and to have effective heuristics to compute good solutions on any graph.

In this paper, we provide a set of three different strategies to solve the problem. We carry out an intensive experimental evaluation of them by comparing the heuristic solutions with the exact solution obtained solving, by means of the solver Cplex, a mathematical formulation that is provided next on the paper (see section 2).

Moreover, in this paper we study a related problem, that is more suitable to model real costs of such location problem on optical networks: finding a spanning tree of a graph such that the degree sum of the branch vertices in the tree is minimized.
Indeed, many devices can only duplicate laser beams, and the effective number of devices to be located on a branch node, in order to replicate lights, is directly related to the number of edges incident to the node. Consider, for example, the generic branch node $j$ in Figure 1(a). In order to transmit the signal, coming from node $i$, to any of the nodes $j_1, j_2, j_3, j_4$, we need to locate on $j$ exactly 3 devices that split the signal, as illustrated in Figure 1(b).

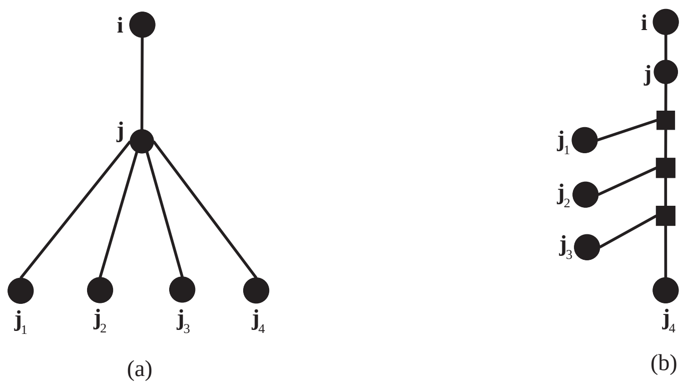

Fig. 1. *Devices that have to be located on the node j.*

That is, at a branch node $j$ with degree $\delta(j)$ we need to locate $\delta(j) - 2$ different devices. Therefore, the related arising problem consists in minimizing the degree sum of the branch vertices of any spanning tree of $G$ (in the sequel this problem will be referred to as MDS).
Let us denote by $q(G)$ the optimum solution of MDS on a connected graph $G$. Note that MBV and MDS are strictly related but are not equivalent. Indeed, the optimum solution $q(G)$ cannot always be directly derived by the optimum solution $s(G)$. Consider for example the simple network in Figure 2, with 8 vertices and 12 edges. The spanning tree with the minimum number of branch vertices is given in Figure 3(a) where vertex 1 is the only vertex with degree greater than two, (i.e., $s(G) = 1$), and equal to $\delta(1) = 7$. However (see Figure 3(b)), the optimum spanning tree for MDS has two branch vertices, i.e., vertices 1 and 4, whose degree sum is equal to $q(G) = 6 < 7$.

This problem is new and not known. In this paper, we address the problem

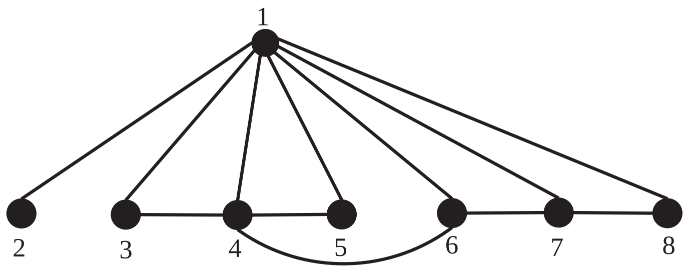

Fig. 2. *MBV and MDS are not equivalent on this network.*

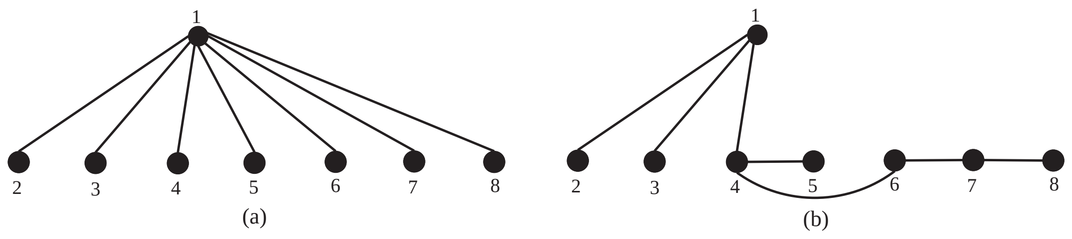

Fig. 3. *(a):Optimal solution of MBV. (b):Optimal solution of MDS.*

by analyzing its complexity. We propose heuristic strategies to solve the problem and evaluate the results by comparing them with the optimum solutions provided by a solver.

The sequel of the paper is organized as follows. Next section contains some complexity results about MDS and the mathematical formulation for both the problems. Section 3 describes the heuristics we developed. In section 4, we analyze the experimental results on an extensive set of scenarios. Conclusions are object of section 5.

## 2 Complexity and Mathematical Formulations

In this section we first clarify the relationship between $s(G)$ and $q(G)$ by stating the NP-completeness of MDS problem and some bounds on the optimal value. Then, we provide a mathematical formulation for both MBV and MDS. We provide a mixed integer single commodity formulation in order to have a polynomial number of constraints (see [6]) and, therefore, to be able to optimally solve, by reasonable computation times, a significant number of instances in order to better evaluate the heuristic results provided by our algorithms.

## *2.1 Complexity results*

Let $G = (V, E)$ be an undirected and connected graph, where $V$ is the vertex set and, $E$ is the edge set. For each $v \in V$, we denote with $\delta_G(v)$ the degree of $v$ in $G$ and with $\Delta(G)$ the maximum degree of a vertex in $G$. We prove now that the problem to decide whether $q(G) \leq k$ is NP-complete. Recall that deciding whether a given graph admits a spanning spider, i.e. $s(G) \leq 1$ is NP-complete [2], and, that given a graph $G$ and a vertex $v$ of $G$ it is NP-complete to decide whether $G$ admits a Hamilton path starting at $v$ [4].

**Theorem 1** *Let $k$ be any fixed positive integer. If $P \neq NP$, then there is no polynomial time algorithm to check whether $q(G) \leq k$.*

*Proof.* Clearly MDS is in NP, because a non-deterministic algorithm needs to guess a spanning tree $T$ of $G$ and verifies whether the sum of the degree of the branch vertices of $T$ is less then or equal to $k$. Since each branch vertex has degree at least 3, then deciding whether $q(G) \leq 5$ reduces to verifies whether $G$ admits a spanning spider. Thus, we may assume $k \geq 6$. Let $G$ be a given graph and $v$ a given vertex of $G$. We build a new graph $H$ by making $k - 2$ copies of $G$ and adding two new vertices $z_1, z_2$ connected by an edge. We then make $z_1$ adjacent to two of the $k - 2$ copies of the vertex $v$, and $z_2$ adjacent to the remaining copies of the vertex $v$. It is easy to check that $H$ admits a spanning tree with branch vertices degree sum at most $k$ if and only if $G$ admits a Hamilton path starting at $v$. $\square$

Obviously, since each branch vertex has a minimum degree of three and a maximum degree equal to $\Delta(G)$, we can state the following straightforward bounds on $q(G)$:

**Proposition 1** $3s(G) \leq q(G) \leq s(G)\Delta(G)$

The lower bound $3s(G)$ is achieved on cubic graphs, on which the problem of computing $q(G)$ is equivalent to computing $s(G)$. Since computing $s(G)$ in NP-complete even on cubic graphs, we have directly the following additional complexity result:

**Theorem 2** *Let $k$ be any fixed positive integer, and $G$ be a cubic graph. If $P \neq NP$, then there is no polynomial time algorithm to check whether $q(G) \leq k$.*

Let us focus first on MBV. In order to define a spanning tree $T$ of $G$ we can send from a source vertex $s \in V$ one unit of flow to every other vertex $v \in V \backslash \{s\}$ of the graph. Although the edges of $G$ are undirected, we define two variables for each edge $e = \{u, v\} \in E$: $f_{uv}$ and $f_{vu}$ that define, respectively, the flow going from $u$ to $v$ and the flow going from $v$ to $u$ along edge $\{u, v\}$. Also, for each edge $e = \{u, v\} \in E$, we consider a binary decisional variable $x_e$ such that $x_e = 1$ when the edge $e$ belongs to $T$, $x_e = 0$ otherwise; finally, for each $v \in V$, we have a binary decisional variable $y_v$ that is equal to 1 if the vertex $v$ is branch, and is equal to 0 otherwise.

Let us denote by $A^+(v) = \{(v, w) \in V \times V | \ \{v, w\} \in E\}$ and $A^-(v) = \{(w, v) \in V \times V \mid \{v, w\} \in E\}$, the set of edges outgoing from $v$ and incoming into $v$ in the directed version of $G$, respectively. The mathematical formulation of MBV is the following:

$$min \sum_{v \in V} y_v \tag{2.1}$$

$$s.t. :$$

$$\sum_{e \in E} x_e = n - 1 \tag{2.2}$$

$$\sum_{(s,v) \in A^+(s)} f_{sv} - \sum_{(v,s) \in A^-(s)} f_{vs} = n - 1 \tag{2.3}$$

$$\sum_{(v,u) \in A^+(v)} f_{vu} - \sum_{(u,v) \in A^-(v)} f_{uv} = -1 \ \forall \ v \ \in V \setminus \{s\} \tag{2.4}$$

$$f_{uv} \le (n-1)x_e \ \forall \ e = \{u, v\} \in E \tag{2.5}$$

$$f_{vu} \le (n-1)x_e \ \forall \ e = \{u, v\} \in E \tag{2.6}$$

$$\sum_{e \in A(v)} x_e - 2 \le (n-1)y_v, \ \forall \ v \ \in V \tag{2.7}$$

$$x_e \in \{0, 1\} \ \forall \ e \ \in E \tag{2.8}$$

$$y_v \in \{0, 1\} \ \forall \ v \ \in V \tag{2.9}$$

$$f_{uv} \ge 0 \ \forall \ e = \{u, v\} \in E \tag{2.10}$$

$$f_{vu} \ge 0 \ \forall \ e = \{u, v\} \in E \tag{2.11}$$

The objective function (2.1) requires to minimize the total number of branches in the tree. Constraints (2.2) ensure that the graph defined by every feasible solution has $n - 1$ edges. Equations (2.3) and (2.4) balance the flow at each vertex and ensure the connectivity of any feasible solution. The constraints (2.5) and (2.6) set the value of each variable $x_e$ equal to 1 whenever at least one between $f_{uv}$ and $f_{vu}$ is positive, that is, when the edge $e$ is selected for the spanning tree. Finally, constraints (2.7) ensure each variable $y_v$ to be equal to 1, whenever $v$ has more than two adjacent edges belonging to the optimum spanning tree.

We emphasize here that we decide to provide such a flow formulation of the problem, since we were interested in being able to solve MBV by means of a solver with reasonable computation times on a large enough set of instances. The study of the geometric properties, and therefore the study of a more "strict" formulation, is out of the scope of this paper, that is instead focused on efficiently solving the problem by means of different heuristic algorithms.

The mathematical formulation for MDS requires the additional integer decisional variables counting the degree of the branch vertices of the solution:

$$z_v = \begin{cases} 0, & \text{if } v \text{ is not branch;} \\ \delta_T(v), & \text{otherwise.} \end{cases} \tag{2.12}$$

The mathematical model for MDS requires to minimize the objective function

$$\sum_{v \in V} z_v \tag{2.13}$$

subject to constraints (2.2)-(2.7) and the additional constraints

$$\sum_{e \in A(v)} x_e - 2 + 2y_v \leq z_v, \ \forall \ v \ \in V. \tag{2.14}$$

Observe that the left side of the above constraints is negative when $y_v$ is 0, that is when vertex $v$ is not branch, therefore $z_v$ is equal to 0 because of the minimization of the objective function. When, on the other hand, a vertex $v$ is branch, $z_v$ is forced to be equal to the left side of (2.14), that is equal to $\delta_T(v)$.

## 3 Different heuristic strategies to solve the problems

In this section we describe the heuristics we developed to solve the problems. We considered three different strategies, namely, the *edge weighting* strategy, the *node coloring* strategy, and, a *combined* strategy.

**Algorithm 1** *Edge weighting strategy*

```
Input: A connected graph $G = (V, E)$
Output: A spanning tree $T(V', E')$
 1: Initialize $T(V', E')$ as $(V, \oslash)$
 2: for all $(u, v) \in E$ do
 3:    $w(u, v) \leftarrow 1$
 4: end for
 5: $A \leftarrow E$
 6: while $|E'| \neq n - 1$ do
 7:    $L \leftarrow \{(u', v') \in A \mid w(u', v') \leq w(u, v),\ \forall \{u, v\} \in A\}$
 8:    $\{u^*, v^*\} \leftarrow select(L)$ {Selection criterion to tie break}
 9:    $A \leftarrow A \setminus \{\{u^*, v^*\}\}$
10:    if $u^*$ and $v^*$ are in different components of $T$ then
11:       $T \leftarrow T \cup \{\{u^*, v^*\}\}$
12:       Update the weights of $u^*$ and $v^*$
13:       $cover(\{u^*, v^*\}, G, T)$
14:    end if
15: end while
```

### *3.1 Edge weighting strategy for MBV*

The idea behind the edge weighting heuristic is to create, from the original unweighted graph $G = (V, E)$, a weighted graph $G' = (V, E, w)$, where $w : E \rightarrow N^+$ is a positive function on the edge set $E$. The weight $w(u, v)$ of the edge $\{u, v\}$ is an hint on the possibility that such an edge could create a branch vertex when selected for a spanning tree.
The algorithm begins by assigning weight 1 to each edge of G. Then, at each iteration, a minimum weight edge among those not in the partial tree $T$, is selected. If the selected edge $\{u^*, v^*\}$ is such that $u^*$ and $v^*$ are in different connected components of $T$, that is $T \cup \{\{u^*, v^*\}\}$ is acyclic, then $\{u^*, v^*\}$ is selected for $T$ and the weight of the edges incident to $u^*$ and $v^*$ is increased by 1. Increasing the weight of the edges incident in $u^*$ and $v^*$ has the effect of making them less desirable for the algorithm selection in the subsequent iterations. The algorithm stops when $n-1$ edges of $G$ are selected. The pseudocode of the algorithm is given next.

Since at each iteration the weight of many edges is increased of the same quantity, we may have multiple edges of minimum weight among which we need to choose. Thus, it has great relevance the rule we adopted to break the ties.

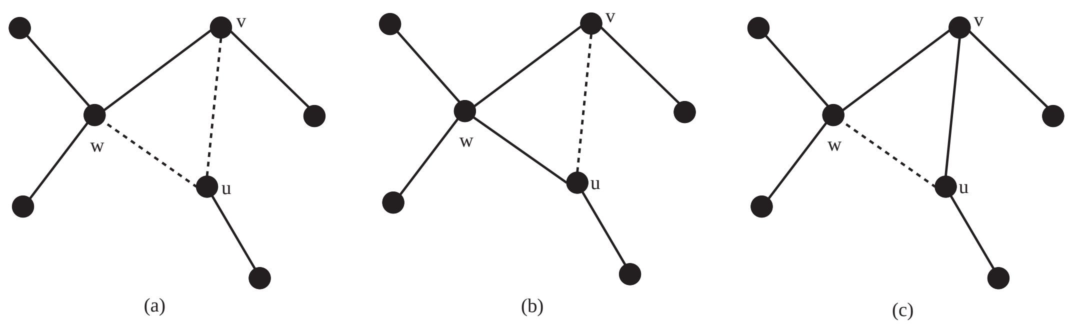

Fig. 4. *Example of the breaking rule for MBV*

Consider the situation in Figure 4(a), where the dashed edges are not yet selected and where we suppose that $\{w, v\}$ and $\{u, v\}$ are the edges of minimum weight at a generic iteration. By selecting $\{w, v\}$ the final spanning tree will have a single branch vertex of degree 4, that is vertex $w$ (see Figure 4(b)). On the other hand, by selecting the edge $\{u, v\}$, the final spanning tree $T$ will have two branch vertices, each of degree 3 (see Figure 4(c)). Then, in order to utilize already branch vertices to cover newly vertices, it is suitable to choose, among edges of the same minimum weight, the one having the endpoints with the maximum degree (vertices $u$ and $w$ in the example). In this way, we try to avoid the creation of *new* branch vertices, attempt to cover the maximum number of vertices, with the edges incident in vertices that are already branch. Finally, observe that during the execution of the algorithm, once a node becomes branch then it is profitable to select the greatest number of its edges. That is, after selecting edge $\{u, v\}$ to be inserted into $T$, if, for example, the endpoint $u$ becomes branch, we examine the edges in $A(u)$ to select those to be inserted directly into $T$, that, obviously, do not create new branch vertices (*cover criterion*). This is carried out by the *cover* procedure.

### *3.2 Node Coloring Strategy for MBV*

Unlike the edge weighting approach, the node coloring strategy assigns to each vertex of the graph a label (color) $c : V \rightarrow$\{G(reen), B(lue), Y(ellow), R(ed)\}. The color $c(v)$ of a vertex $v$ indicates if such a vertex becomes branch when an edge incident to it is selected to be inserted into the spanning tree. The label is assigned to each vertex according to (3.1).

$$c(v) = \begin{cases} G, \text{ if } d_T(v) = 0; \\ B, \text{ if } d_T(v) = 1; \\ Y, \text{ if } d_T(v) = 2; \\ R, \text{ if } d_T(v) \geq 3. \end{cases} \tag{3.1}$$

The algorithm starts with the initial forest $T(V', E') = (V, \emptyset)$ and assigns the green color to each vertex $v \in V'$. At each iteration an edge is selected to be inserted into the spanning tree according to the colors of its endpoints. Note that the number of the new branch vertices created after the insertion of edge $\{u, v\}$ is the number of its yellow endpoints. Thus, the algorithm selects the edge with the minimum number of yellow endpoints. Also, in order to minimize the number of new potential yellow vertices (that can become branches), ties are broken by choosing the edge with the minimum number of blue endpoints. The algorithm stops when $n-1$ edges are selected. Obviously, also in this case is called the cover procedure when a node becomes branch.

Let $n_Y(u, v)$ and $n_B(u, v)$ be, respectively, the number of yellow and blue endpoints of the edge $(u, v)$ we formalize the algorithm as follows:

**Algorithm 2** Node coloring heuristic

```
Input: A graph $G = (V, E)$
Output: A spanning tree $T(V', E')$
 1: Initialize $T(V', E')$ as $(V, \oslash)$
 2: for all $v \in V$ do
 3:     $c(v) \leftarrow G$
 4: end for
 5: $A \leftarrow E$
 6: while $|E'| \neq n-1$ do
 7:     $L \leftarrow \{\{u', v'\} \in A \mid n_Y(u', v') \leq n_Y(u, v), \ \forall \{u', v'\} \in A\}$
 8:     $\{u, v\} \leftarrow arg \min_{\{u,v\} \in L} \{ n_B(u, v) \}$
 9:     $A \leftarrow A \setminus \{(u, v)\}$
10:     if $u$ and $v$ are in different components of $T$ then
11:         $T \leftarrow T \cup \{\{u, v\}\}$
12:         Updates the color of $u$ and $v$
13:         $cover(\{u, v\}, G, T)$
14:     end if
15: end while
```

## *3.3 Combined approach for MBV*

A combined approach is also implemented by considering both the two criteria above mentioned. In particular, we decided to both assign weights to edges according to the edge weighting strategy and labels to vertices according to the node coloring strategy. The combined approach selects at each iteration the edge with minimum weight and breaks the ties by applying first the minimum blue criterion and then the maximum degree criterion.

### *3.3.1 Edge weighting, Node coloring and Combined approaches for MDS*

The three resolution proposed strategies are suitable to solve MDS too. There are, however, two main differences to take care of when applying such strategies to solve MDS:

- obviously, the cover criterion after a vertex becomes branch is not applied;
- the tie breaking rule for MDS, based on the degree of the endpoints of the edge, consists in selecting, among the edges of the same minimum weight, the one whose endpoints have minimum degree, in order to try not to increase the degree of the newly branch vertices.

## 4 Computational Results

In this section we describe the experimentation performed to evaluate the solutions quality of our heuristics.
In order to create problem instances we used three different problem generators. In particular, we use the generators *Netgen* (see [7]), *Genmax* and *Random*, all available by the ftp service from the host of the DIMACS, *Center for Discrete Mathematics and Theoretical Computer Science. Netgen* and *Genmax* are network flow problem generators, that generate instances with random edges and uniform capacity. We adapted the instances obtained by these generators to our problem, discarding the edge capacity and transforming the network from directed to undirected.

For each generator, we considered 8 different size for the total number of the vertices of the graph: $n = 20, 30, 40, 50, 100, 300, 500, 1000$. Fixed the value of $n$, we considered 5 different values of density (the ratio between the number of edges and the number of vertices): $d = 1.5, 2, 4, 10, 15$. Thus, for each generator, we considered 40 different scenarios. For a given scenario, we consider 5 different problem instances belonging to that scenario, so that the report values are the average values obtained considering the 5 instances.
As mentioned before, in order to obtain solution values that we could compare with the heuristic ones, we use the commercial mixed integer programming software CPLEX to get either the exact solution of a given problem instance (when this is reached in at most three hours) or a lower bound on the exact value when the imposed time limit is reached. We denote this last case with the term *dnf* (did not finish) in the column relative to the execution time, and the lower bound is denoted with a " * " in the corresponding column.

Tables 1-3 summarize results, respectively for Netgen, Maxgen and Random instances. Each table is divided into two parts: the first part is relative to MBV problem and the second one to the MDS problem. For each problem we give

(i) the exact solution value provided by cplex and its running time and (ii) the solution values provided by the heuristics Edge Weighting (E.W. column), Node Coloring (N.C. column) and Combined Approach (C.A. column).

Computational times of the heuristics are negligible and are not reported. The Combined approach always gives a better result than both the Edge Weighting approach and the Node Coloring one, when applied to solve the *MBV* problem. The solution value is optimum for 22 different scenarios out of 99 (among those for which Cplex returns an optimum solution), and, in the worst case it returns a value that is at most 3 times the optimum one. The Edge Weighting approach is, on the other hand, always the best among the three proposed approach in solving the MDS problem. It finds the optimum value for 14 scenarios out of 75 (among those for which Cplex returns the optimum). The worst case value is at most 4 times the optimum value and it is achieved in only two scenarios: the random graph with $n = 100$ and $d = 4$ (see Table 3) and the graph generated by the maxgen generator when $n = 100$ and $d = 15$ (see Table 2). For all the other instances the solution value is such that, on the average, the distance between the optimum solution and the heuristic one is less than 0.37%.

Table 1: Test results on the Netgen instances.

| *Netgen instances* | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Scenario | | MBV | | | | | MDS | | | | |
| | | CPLEX | | HEURISTICS | | | CPLEX | | HEURISTICS | | |
| *n* | *d* | *time (sec.)* | *value* | E.W. | N.C. | C.A. | *time (sec.)* | *value* | E.W. | N.C. | C.A. |
| 20 | 1.5 | 0.024 | 0.4 | 0.6 | 0.5 | 0.4 | 0.03 | 1.4 | 1.4 | 1.6 | 1.8 |
| 20 | 2.0 | 0.058 | 0.4 | 0.6 | 0.6 | 0.5 | 0.03 | 1.6 | 1.8 | 3.0 | 2.4 |
| 20 | 4.0 | 0.204 | 0.2 | 0.6 | 0.4 | 0.4 | 0.35 | 1.2 | 2.4 | 3.0 | 2.8 |
| 20 | 10.0 | 0.307 | 0.2 | 0.4 | 0.4 | 0.2 | 0.25 | 1.6 | 1.8 | 2.4 | 2.0 |
| 20 | 15.0 | 0.781 | 0.4 | 0.8 | 0.6 | 0.4 | 0.17 | 1.6 | 1.6 | 1.8 | 1.8 |
| 30 | 1.5 | 0.076 | 1.2 | 2.2 | 1.8 | 1.4 | 0.05 | 3.8 | 4.0 | 4.2 | 4.2 |
| 30 | 2.0 | 0.298 | 0.4 | 1.0 | 0.6 | 0.6 | 0.22 | 2.2 | 2.6 | 5.2 | 4.2 |
| 30 | 4.0 | 0.556 | 0.4 | 1.0 | 0.6 | 0.6 | 0.15 | 2.2 | 2.4 | 3.0 | 2.8 |
| 30 | 10.0 | 0.562 | 0.2 | 1.0 | 0.6 | 0.4 | 0.66 | 0.6 | 0.8 | 1.2 | 1.0 |
| 30 | 15.0 | 0.620 | 0.4 | 0.8 | 0.8 | 0.6 | 0.71 | 1.4 | 1.6 | 2.0 | 2.0 |
| 40 | 1.5 | 0.236 | 1.4 | 2.0 | 2.0 | 1.8 | 0.07 | 4.2 | 4.6 | 5.2 | 4.8 |
| 40 | 2.0 | 0.800 | 0.2 | 0.4 | 0.6 | 0.2 | 0.27 | 6.2 | 7.2 | 9.0 | 8.4 |
| 40 | 4.0 | 1.474 | 0.8 | 1.2 | 0.8 | 0.8 | 0.46 | 2.6 | 3.4 | 3.4 | 3.4 |
| 40 | 10.0 | 2.986 | 0.2 | 0.4 | 0.4 | 0.4 | 0.51 | 1.2 | 1.6 | 3.0 | 1.8 |
| 40 | 15.0 | 3.384 | 0.2 | 0.8 | 0.8 | 0.4 | 3.26 | 0.6 | 0.8 | 1.2 | 1.0 |
| 50 | 1.5 | 0.210 | 0.6 | 2 | 1.8 | 1.6 | 0.50 | 2.8 | 3.0 | 3.4 | 3.2 |
| 50 | 2.0 | 0.902 | 0.2 | 1.2 | 0.8 | 0.4 | 1.04 | 2.2 | 2.8 | 4.0 | 4.0 |
| 50 | 4.0 | 2.888 | 0.8 | 1.6 | 1.2 | 1.0 | 3.34 | 0.4 | 1.0 | 3.4 | 3.4 |
| 50 | 10.0 | 5.674 | 0.2 | 0.4 | 0.4 | 0.2 | 6.82 | 0.8 | 1.2 | 2.8 | 2.6 |
| 50 | 15.0 | 6.862 | 0.2 | 0.6 | 0.8 | 0.4 | 8.35 | 0.6 | 0.8 | 1.2 | 1.0 |
| 100 | 1.5 | 0.712 | 0.6 | 2.8 | 1.4 | 1.6 | 2.57 | 2.8 | 3.4 | 4 | 3.8 |
| 100 | 2.0 | 9.816 | 0.8 | 2.3 | 1.6 | 1.4 | 5.52 | 2.4 | 2.8 | 4 | 4 |
| 100 | 4.0 | 22.210 | 0.6 | 1.4 | 1.0 | 0.8 | 7.22 | 3.8 | 4.2 | 5.6 | 5.2 |
| 100 | 10.0 | 54.608 | 0.4 | 1.4 | 1.4 | 0.8 | 8.51 | 3.6 | 4.2 | 4.2 | 4.4 |
| 100 | 15.0 | 83.920 | 0.2 | 0.4 | 0.4 | 0.4 | 9.18 | 1.6 | 1.8 | 3 | 2.8 |
| 300 | 1.5 | 21.520 | 1.2 | 2.0 | 1.4 | 1.4 | dnf | *0.0 | 3.8 | 4.0 | 4.0 |
| 300 | 2.0 | 104.722 | 1.4 | 2.4 | 1.6 | 1.4 | dnf | *0.0 | 4.6 | 4.8 | 4.6 |
| 300 | 4.0 | 304.670 | 1.0 | 2.0 | 1.2 | 1.0 | dnf | *0.0 | 3.2 | 5.2 | 5.0 |
| 300 | 10.0 | dnf | *1.4 | 3.4 | 2.2 | 1.8 | dnf | *0.0 | 4.8 | 4.8 | 4.6 |
| 300 | 15.0 | dnf | *1.2 | 2.6 | 1.4 | 1.4 | dnf | *0.0 | 4.6 | 5.2 | 5.0 |
| 500 | 1.5 | 154.200 | 1.4 | 2.2 | 1.2 | 1.4 | dnf | *0.0 | 4.8 | 5.6 | 5.4 |
| 500 | 2.0 | 211.837 | 1.2 | 1.8 | 1.6 | 1.4 | dnf | *0.0 | 6.6 | 7.2 | 7.0 |
| 500 | 4.0 | dnf | *0.8 | 2.0 | 1.4 | 1.0 | dnf | *0.0 | 4.4 | 8.2 | 8.2 |
| 500 | 10.0 | dnf | *2.2 | 3.8 | 3.4 | 2.4 | dnf | *0.0 | 7.4 | 8.2 | 8.2 |
| 500 | 15.0 | dnf | *2.4 | 5.0 | 3.2 | 2.6 | dnf | *0.0 | 7.6 | 8.2 | 8.2 |
| 1000 | 1.5 | 329.860 | 2.2 | 3.6 | 2.4 | 2.4 | dnf | *0.0 | 6.6 | 7.8 | 7.6 |
| 1000 | 2 | dnf | *1.8 | 3.2 | 2.8 | 2.0 | dnf | *0.0 | 8.6 | 9.6 | 9.4 |
| 1000 | 4 | dnf | *2.4 | 5.6 | 3.8 | 3.4 | dnf | *0.0 | 7.8 | 8.4 | 8.2 |
| 1000 | 10 | dnf | *2.6 | 4.0 | 3.0 | 2.8 | dnf | *0.0 | 8.2 | 8.8 | 8.8 |
| 1000 | 15 | dnf | *1.2 | 2.6 | 2.2 | 1.8 | dnf | *0.0 | 4.4 | 5.2 | 5.2 |

Table 2: Test results on the Maxgen instances.

| *Maxgen instances* | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Scenario | | MBV | | | | | MDS | | | | |
| | | CPLEX | | HEURISTICS | | | CPLEX | | HEURISTICS | | |
| $n$ | $d$ | *time (sec.)* | *value* | E.W. | N.C. | C.A. | *time (sec.)* | *value* | E.W. | N.C. | C.A. |
| 20 | 1.5 | 0.010 | 1.2 | 3.0 | 3.0 | 2.0 | 0.02 | 4.2 | 5.4 | 6.2 | 6.0 |
| 20 | 2.0 | 0.048 | 0.2 | 1.8 | 2.8 | 0.8 | 0.05 | 2.4 | 4.2 | 7.2 | 7.0 |
| 20 | 4.0 | 0.080 | 0.0 | 0.0 | 0.0 | 0.0 | 0.10 | 0.0 | 0.0 | 0.0 | 0.0 |
| 20 | 10.0 | 0.124 | 0.0 | 0.0 | 0.0 | 0.0 | 0.13 | 0.0 | 0.0 | 0.0 | 0.0 |
| 20 | 15.0 | 0.235 | 0.0 | 0.0 | 0.0 | 0.0 | 0.34 | 0.0 | 0.0 | 0.0 | 0.0 |
| 30 | 1.5 | 0.060 | 1.2 | 3.0 | 3.0 | 1.4 | 0.36 | 6.2 | 8.2 | 9.4 | 9.2 |
| 30 | 2.0 | 0.076 | 0.8 | 2.4 | 2.0 | 1.2 | 0.49 | 3.8 | 4.2 | 4.4 | 4.2 |
| 30 | 4.0 | 0.092 | 0.6 | 1.0 | 1.0 | 1.0 | 1.15 | 2.4 | 3.2 | 4.2 | 4.2 |
| 30 | 10.0 | 1.602 | 0.0 | 0.4 | 0.2 | 0.0 | 1.26 | 0.0 | 1.0 | 1.2 | 1.0 |
| 30 | 15.0 | 2.202 | 0.0 | 0.6 | 0.4 | 0.2 | 2.50 | 0.0 | 0.6 | 0.8 | 0.8 |
| 40 | 1.5 | 0.360 | 2.2 | 6.0 | 5.0 | 2.4 | 0.18 | 9.2 | 12.2 | 13.8 | 13.2 |
| 40 | 2.0 | 0.756 | 1.0 | 4.0 | 3.0 | 1.2 | 0.23 | 3.4 | 7.2 | 9.2 | 9.0 |
| 40 | 4.0 | 1.608 | 0.8 | 2.4 | 2.0 | 1.2 | 0.56 | 3.4 | 6.2 | 8.4 | 8.2 |
| 40 | 10.0 | 1.992 | 0.2 | 1.0 | 0.8 | 0.4 | 1.38 | 2.4 | 3.2 | 4.2 | 4.0 |
| 40 | 15.0 | 2.826 | 0.0 | 0.8 | 0.0 | 0.0 | 2.19 | 0.0 | 0.6 | 0.4 | 0.6 |
| 50 | 1.5 | 0.270 | 2.8 | 7.0 | 6.0 | 3.2 | 0.10 | 10.2 | 13.6 | 15.2 | 15.0 |
| 50 | 2.0 | 2.052 | 1.4 | 6.0 | 5.0 | 2.0 | 0.77 | 4.4 | 5.6 | 7.2 | 7.0 |
| 50 | 4.0 | 3.936 | 0.6 | 2.0 | 3.0 | 1.2 | 1.26 | 2.4 | 4.2 | 6.4 | 6.2 |
| 50 | 10.0 | 4.522 | 0.0 | 0.0 | 0.0 | 0.0 | 5.12 | 0.0 | 0.6 | 0.6 | 0.4 |
| 50 | 15.0 | 6.743 | 0.0 | 0.0 | 0.0 | 0.0 | 6.79 | 0.0 | 0.0 | 0.0 | 0.0 |
| 100 | 1.5 | 14.324 | 7.2 | 15.0 | 12.0 | 8.2 | 7.04 | 34.2 | 39.2 | 43.0 | 42.2 |
| 100 | 2.0 | 37.814 | 1.6 | 5.4 | 4.2 | 2.4 | 11.37 | 11.6 | 14.4 | 17.2 | 17.2 |
| 100 | 4.0 | 49.068 | 1.4 | 4.0 | 3.0 | 2.2 | 17.29 | 5.2 | 5.8 | 6.4 | 6.2 |
| 100 | 10.0 | 53.764 | 0.0 | 1.0 | 1.0 | 0.2 | 24.65 | 0.0 | 0.6 | 0.6 | 0.4 |
| 100 | 15.0 | 99.592 | 0.2 | 1.0 | 0.8 | 0.4 | 28.88 | 0.6 | 3.0 | 3.2 | 3.2 |
| 300 | 1.5 | dnf | *4.8 | 12.0 | 9.0 | 6.8 | dnf | *37.178 | 42.8 | 44.6 | 44.4 |
| 300 | 2.0 | dnf | *5.6 | 12.4 | 10.2 | 7.2 | dnf | *39.329 | 44.2 | 46.8 | 46.6 |
| 300 | 4.0 | dnf | *0.0 | 8.2 | 4.2 | 3.4 | dnf | *0.0 | 5.4 | 7.2 | 7.0 |
| 300 | 10.0 | dnf | *0.0 | 7.0 | 10.0 | 4.6 | dnf | *0.0 | 4.2 | 5.8 | 5.6 |
| 300 | 15.0 | dnf | *0.0 | 3.0 | 4.0 | 2.4 | dnf | *0.0 | 2.4 | 4.6 | 4.4 |
| 500 | 1.5 | 211.690 | 2.4 | 12.0 | 8.2 | 3.8 | dnf | *0.0 | 4.2 | 6.8 | 6.8 |
| 500 | 2.0 | dnf | *0.0 | 7.2 | 7.0 | 4.4 | dnf | *0.0 | 3.6 | 4.8 | 4.6 |
| 500 | 4.0 | dnf | *0.0 | 8.6 | 8.4 | 5.2 | dnf | *0.0 | 1.6 | 2.4 | 2.2 |
| 500 | 10.0 | dnf | *0.0 | 8.0 | 7.0 | 4.8 | dnf | *0.0 | 0.8 | 1.2 | 1.2 |
| 500 | 15.0 | dnf | *0.0 | 6.0 | 7.4 | 5.4 | dnf | *0.0 | 1.4 | 2.4 | 2.4 |
| 1000 | 1.5 | dnf | *0.0 | 3.6 | 2.8 | 1.2 | dnf | *0.0 | 7.2 | 9.4 | 9.2 |
| 1000 | 2.0 | dnf | *1.6 | 8.8 | 5.6 | 3.2 | dnf | *0.0 | 5.2 | 6.8 | 6.6 |
| 1000 | 4.0 | dnf | *0.6 | 7.0 | 5.0 | 1.8 | dnf | *0.0 | 3.4 | 4.6 | 4.4 |
| 1000 | 10.0 | dnf | *0.0 | 3.2 | 2.4 | 1.2 | dnf | *0.0 | 2.4 | 3.8 | 3.8 |
| 1000 | 15.0 | dnf | *0.0 | 2.0 | 1.4 | 0.8 | dnf | *0.0 | 1.2 | 2.6 | 2.4 |

Table 3: Test results on the Random instances.

| *Random instances* | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Scenario | | MBV | | | | | MDS | | | | |
| | | CPLEX | | HEURISTICS | | | CPLEX | | HEURISTICS | | |
| *n* | *d* | *time (sec.)* | *value* | E.W. | N.C. | C.A. | *time (sec.)* | *value* | E.W. | N.C. | C.A. |
| 20 | 1.5 | 0.094 | 1.2 | 2.4 | 2.8 | 1.8 | 0.02 | 4.2 | 7.2 | 12 | 11.8 |
| 20 | 2.0 | 0.166 | 0.6 | 1.8 | 1.2 | 0.8 | 0.05 | 2.6 | 4.2 | 7.2 | 6.8 |
| 20 | 4.0 | 0.198 | 0.0 | 0.4 | 0.4 | 0.2 | 0.10 | 0.0 | 0.0 | 0.6 | 0.4 |
| 20 | 10.0 | 0.212 | 0.0 | 0.0 | 0.0 | 0.0 | 0.13 | 0.0 | 0.0 | 0.6 | 0.4 |
| 20 | 15.0 | 0.404 | 0.0 | 0.0 | 0.0 | 0.0 | 0.34 | 0.0 | 0.0 | 0.0 | 0.0 |
| 30 | 1.5 | 0.496 | 1.8 | 3.4 | 4.6 | 2.2 | 0.36 | 5.6 | 6.2 | 7.4 | 7.2 |
| 30 | 2.0 | 0.966 | 2.4 | 3.8 | 3.8 | 2.6 | 0.49 | 7.4 | 7.6 | 7.6 | 7.4 |
| 30 | 4.0 | 1.342 | 0.2 | 0.4 | 0.4 | 0.2 | 1.15 | 3.0 | 3.2 | 4.2 | 4.0 |
| 30 | 10.0 | 2.432 | 0.0 | 0.0 | 0.0 | 0.0 | 1.26 | 0.0 | 0.0 | 0.0 | 0.0 |
| 30 | 15.0 | 3.212 | 0.0 | 0.0 | 0.0 | 0.0 | 2.50 | 0.0 | 0.0 | 0.0 | 0.0 |
| 40 | 1.5 | 1.074 | 2.6 | 5.8 | 5.6 | 3.2 | 0.18 | 9.2 | 11.2 | 13.0 | 12.8 |
| 40 | 2.0 | 2.466 | 0.8 | 3.2 | 3.8 | 1.4 | 0.23 | 3.4 | 4.2 | 9.2 | 8.8 |
| 40 | 4.0 | 3.608 | 1.6 | 2.4 | 2.2 | 2.2 | 0.56 | 5.4 | 7.8 | 10.2 | 9.8 |
| 40 | 10.0 | 4.213 | 0.2 | 0.6 | 0.6 | 0.4 | 1.38 | 3.0 | 3.6 | 6.6 | 6.0 |
| 40 | 15.0 | 5.230 | 0.0 | 0.0 | 0.0 | 0.0 | 2.19 | 0.0 | 0.0 | 0.0 | 0.0 |
| 50 | 1.5 | 3.306 | 3.6 | 6.0 | 7.4 | 4.6 | 0.10 | 10.8 | 13.6 | 15.2 | 14.8 |
| 50 | 2.0 | 3.852 | 1.8 | 4.8 | 4.8 | 2.2 | 0.77 | 5.6 | 5.6 | 7.2 | 6.8 |
| 50 | 4.0 | 3.936 | 0.8 | 2.0 | 2.4 | 1.2 | 1.26 | 4.2 | 5.2 | 6.2 | 5.8 |
| 50 | 10.0 | 4.288 | 0.2 | 0.4 | 0.4 | 0.4 | 5.12 | 3.2 | 4.4 | 5.2 | 4.8 |
| 50 | 15.0 | 5.754 | 0.0 | 0.0 | 0.0 | 0.0 | 6.79 | 0.0 | 0.0 | 0.0 | 0.0 |
| 100 | 1.5 | 180.638 | 7.6 | 10.8 | 10.8 | 8.8 | 7.04 | 34.2 | 39.2 | 43 | 40.2 |
| 100 | 2.0 | 185.348 | 2.2 | 5.6 | 4.2 | 3.2 | 11.37 | 11.8 | 14.4 | 17.2 | 16.8 |
| 100 | 4.0 | 190.314 | 0.0 | 4.6 | 3.4 | 1.2 | 17.29 | 0.0 | 4.2 | 6 | 4.8 |
| 100 | 10.0 | 274.604 | 0.0 | 0.6 | 1.0 | 0.4 | 24.65 | 0.0 | 3.4 | 4.2 | 3.8 |
| 100 | 15.0 | 297.134 | 0.0 | 0.4 | 0.4 | 0.4 | 28.88 | 0.0 | 3.2 | 4.4 | 4.0 |
| 300 | 1.5 | 118.000 | 20.4 | 29.2 | 28.6 | 21.8 | dnf | *37.178 | 62.8 | 64.6 | 62.4 |
| 300 | 2.0 | 187.005 | 2.2 | 6.4 | 5.2 | 4.0 | dnf | *39.329 | 44.2 | 46.8 | 44.8 |
| 300 | 4.0 | 287.105 | 1.2 | 4.0 | 2.4 | 2.2 | dnf | *0.0 | 5.4 | 7.2 | 6.8 |
| 300 | 10.0 | dnf | *0.0 | 4.8 | 7.6 | 4.6 | dnf | *0.0 | 4.2 | 5.8 | 5.4 |
| 300 | 15.0 | dnf | *0.0 | 3.6 | 5.0 | 3.2 | dnf | *0.0 | 2.4 | 4.6 | 4.2 |
| 500 | 1.5 | dnf | *4.4 | 11.0 | 8.2 | 5.8 | dnf | *0.0 | 14.2 | 16.8 | 16.2 |
| 500 | 2.0 | dnf | *0.4 | 4.4 | 3.4 | 2.4 | dnf | *0.0 | 3.6 | 4.8 | 4.4 |
| 500 | 4.0 | dnf | *0.0 | 6.4 | 6.0 | 2.2 | dnf | *0.0 | 1.6 | 2.4 | 2.2 |
| 500 | 10.0 | dnf | *0.0 | 5.4 | 4.2 | 2.8 | dnf | *0.0 | 0.8 | 1.2 | 1.0 |
| 500 | 15.0 | dnf | *0.0 | 5.6 | 6.8 | 2.4 | dnf | *0.0 | 1.4 | 2.4 | 2.0 |
| 1000 | 1.5 | dnf | *8.8 | 14.6 | 12.8 | 10.2 | dnf | *0.0 | 27.2 | 29.4 | 28.8 |
| 1000 | 2.0 | dnf | *0.6 | 5.2 | 4.8 | 3.2 | dnf | *0.0 | 5.2 | 6.8 | 6.6 |
| 1000 | 4.0 | dnf | *0.8 | 6.2 | 6.4 | 3.8 | dnf | *0.0 | 3.4 | 4.6 | 4.4 |
| 1000 | 10.0 | dnf | *0.0 | 5.4 | 4.8 | 2.4 | dnf | *0.0 | 2.4 | 3.8 | 3.6 |
| 1000 | 15.0 | dnf | *0.0 | 4.2 | 2.6 | 1.2 | dnf | *0.0 | 1.2 | 2.6 | 2.2 |

## 5 Conclusions

In this paper we addressed two combinatorial problems that have a great relevance in the field of optical networks design. In particular, given a graph $G$, the first problem consists in finding a spanning tree of $G$ with the minimum number of branch vertices, while the second problem consists in finding a spanning tree of $G$ such that the degree sum of its branch vertices is minimized.

The two problems are quite new. We analyzed their relation and provide several heuristics to solve them. Our computational results show the effectiveness of the proposed approaches, that return solutions near to the optimum ones provided by a solver, in negligible computational times.

## References

[1] H. Broesma, X. Li. *Spanning Trees with many or few colors in in edge-colored graphs.* Discussiones Mathematicae Graph Theory, 17, 259-269, 1997.

[2] L. Gargano, P. Hell, L. Stacho and U. Vaccaro. *Spanning trees with bounded number of branch vertices.* ICALP' 02, Malága, Spain 2002.

[3] R. Cerulli, A. Fink, M. Gentili, S. Vo$\beta$, *Metaheuristics comparison for the minimum labelling spanning tree problem.* In: B.L. Golden, S. Raghavan and E.A. Wasil (eds.), The Next Wave on Computing, Optimization, and Decision Technologies, Springer, New York (2005), 93 - 106. [ISBN: 0-387-23528-0].

[4] R.M. Karp. *Reducibility among combinatorial problems.* Complexity of Computer C, R.E. Miller and J.W. Thatcher (eds.), Plenum Press, (1972), 82-103.

[5] P. Klein, R.Ravi. *A Nearly Best-Possible Algorithm for Node-Weighted Steiner Trees.* Journal of Algorithms 19, 104-115 (1995).

[6] T. L. Magnanti and L. A. Wolsey, Optimal trees,Network Models (M. O. Ball, T. L. Magnanti, C. L. Monma, and Nemhauser G. L., eds.), Handbooks in Operations Research and Management Science, vol. 7, North Holland, 1995, pp. 503–615.

[7] Klingman, D., A. Napier, and J. Stutz, *"NETGEN: A Program for Generating Large Scale Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems*, Management Science 20, 5, 814-821 (1974)

[8] D. S. Johnson and C. C. McGeoch, editors. *Network Flows and Matching: First DIMACS Implementation Challenge.* AMS, 1993.

# EXPERIMENTAL COMPARISON OF ALGORITHMS FOR BOUNDED-DEGREE SPANNING TREE PROBLEM

R. Cerulli, M. Gentili, A. Iossa

***Keywords:***

Spanning Trees
Spanning Spider
Branch Vertices
Heuristics
Optical Networks

# EXPERIMENTAL COMPARISON OF ALGORITHMS FOR BOUNDED-DEGREE SPANNING TREE PROBLEM

Corresponding author:

Monica Gentili
Dipartimento di Matematica ed Informatica
Universitá di Salerno
Via Ponte Don Melillo, 84084, Fisciano (SA)
Tel. +39 089 963326
Fax +39 089 963303
email: mgentili@unisa.it