European Journal of Operational Research 232 (2014) 442–453

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Relations, models and a memetic approach for three degree-dependent spanning tree problems

C. Cerrone [a], R. Cerulli [b], A. Raiconi [b,*]

[a] Department of Computer Science, University of Salerno, Via Giovanni Paolo II n. 132, 84084 Fisciano (SA), Italy
[b] Department of Mathematics, University of Salerno, Via Giovanni Paolo II n. 132, 84084 Fisciano (SA), Italy

ARTICLE INFO

Article history:
Received 9 May 2012
Accepted 22 July 2013
Available online 29 July 2013

Keywords:
Combinatorial optimization
Memetic algorithms
Spanning trees
Optical networks

ABSTRACT

In this paper we take into account three different spanning tree problems with degree-dependent objective functions. The main application of these problems is in the field of optical network design. In particular, we propose the classical Minimum Leaves Spanning Tree problem as a relevant problem in this field and show its relations with the Minimum Branch Vertices and the Minimum Degree Sum Problems. We present a unified memetic algorithm for the three problems and show its effectiveness on a wide range of test instances.

1. Introduction

Spanning tree problems with degree-related objective functions or constraints are widely studied in the field of network design. In such problems, we generally look for spanning trees that optimize or respect certain properties related to the degree of the vertices, in order to model cost factors or restrictions deriving from the underlying real-world applications.

In the *Minimum Branch Vertices Problem* (or MBV), we look for the spanning tree with the minimum number of vertices (called *branch* vertices) with a degree higher than two.

This problem has great relevance, for instance, in the context of multicast on optical networks. On such networks, the optical signal can be split and therefore sent from a source to multiple destinations by using an appropriate network device (*switch*). Multicast communications can therefore be performed through a spanning tree of the network (light-tree), by placing a switch on each branch vertex. For several reasons (such as budget constraints or signal quality preservation, among others) it can be important to determine the spanning tree which requires the minimum number of switches, that is, the optimal solution for the MBV problem.

Indeed, many switch devices can only duplicate laser beams; therefore, the actual number of devices to be located on a branch vertex is related to the degree of the node. For this reason, the problem of minimizing the degree sum of the branch vertices of any spanning tree of the network (*Minimum Degree Sum Problem* or MDS) has been proposed in the literature.

However, as can be noted from Fig. 1, if $\delta(u, T)$ is the degree of a branch node that is used to propagate information, the exact number of required devices is $\delta(u, T) - 2$. More in general, in this context, the optimization problem consists in minimizing the degree sum of the branch vertices less the cardinality of the set of branch vertices multiplied by two. In this paper we show for the first time that this problem, which models the considered underlying application more accurately than MDS, is equivalent to the well known *Minimum Leaves Problem* (ML), i.e. the problem of finding the spanning tree with the minimum number of degree-1 vertices. This will be proved in Section 2.

The aim of this paper is therefore to propose ML as a relevant problem in the field of optical network design and to show that it is closely related to MDS and MBV, by demonstrating some theoretical properties linking their three objective functions. These objectives are also pursued by presenting a unified memetic algorithm that makes use of a single set of rules to perform crossover, mutation and local search operations for the three problems. An extensive experimental analysis proves the effectivity of the proposed approach.

MBV has been first introduced in Gargano, Hell, Stacho, and Vaccaro (2002) where the problem was shown to be $\mathcal{NP}$-Hard. In Carrabs, Cerulli, Gaudioso, and Gentili (in press) the authors present four different mathematical formulations and compare the results of different relaxations, solving the lagrangian dual by means of a standard subgradient method and an ad hoc finite ascent algorithm. In Gargano and Hammar (2003) and Gargano, Hammar, Hell, Stacho, and Vaccaro (2004), the authors give

* Corresponding author. Tel.: +39 089963385.
E-mail addresses: ccerrone@unisa.it (C. Cerrone), raffaele@unisa.it (R. Cerulli), araiconi@unisa.it (A. Raiconi).

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.07.029

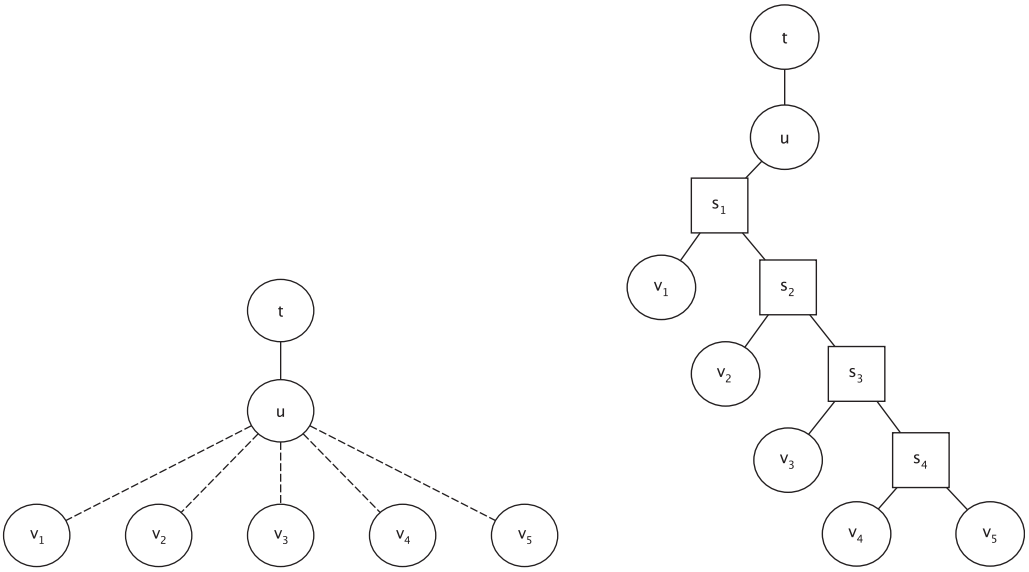


Fig. 1. Example subtree. Switches $s_1,\ldots,s_4$ need to be located on node u to transmit information from t to $v_1,\ldots,v_5$.

conditions for the existence of *spanning spiders* (i.e. spanning tree containing at most one branch vertex) and, more in general, spanning trees with a bounded number of branch vertices. Another problem related to MBV is the *degree-constrained minimum spanning tree*. Given an edge-weighted graph $G = (V, E)$ and a value $b(i) \geqslant 1$, $\forall i \in V$, the aim is to find a spanning tree with minimum weight such that the degree of each node i is bounded by $b(i)$. In Ribeiro and de Souza (2002), the authors solve the problem by presenting a VNS metaheuristic that embeds a Variable Neighborhood Descent (VND) strategy for local search. An improvement of this algorithm is described in de Souza and Martins (2008). The new algorithm uses guiding strategies based on the Second Order algorithm in the shaking phase and the Skewed approach to avoid degeneration into a multistart heuristic. A branch-and-cut method for solving the problem is discussed in Caccetta and Hill (2001). In Duhamel, Gouveia, Moura, and Souza (2011), the authors present a generalization of the problem which considers a non-linear stepwise cost function on every node. They present two linear programming formulations as well as a hybrid GRASP/VND metaheuristic embedding a Path Relinking strategy applied at the end of each GRASP iteration. MDS has been presented and analyzed in Cerulli, Gentili, and Iossa (2009), which also contains some mathematical formulations and heuristic procedures for both MBV and MDS. Other heuristic approaches for MBV and MDS have been recently proposed in Sundar, Singh, and Rossi (2012). The ML problem was proven to be $\mathcal{NP}$-Hard and hard to approximate in Lu and Ravi (1996). In Salamon and Wiener (2008), the authors introduce some approximation algorithms for the related problem of maximizing the number of internal nodes (of course, the two problems have identical optimal solutions). In Fernandes and Gouveia (1998), given an edge-weighted graph and a natural number $k \geqslant 1$, the authors study the problem of finding the minimum weight spanning tree with exactly k leaves. Two mathematical formulations derived from the minimum weight spanning tree problem as well as upper bounding and lower bounding schemes are presented.

The sequel of the paper is organized as follows. Section 2 contains the formal definition of the studied problems, and the demonstrations of several relations among them. Section 3 introduces some mathematical formulations for each of the three problems, and Section 4 describes the memetic algorithm that we propose to solve them. Section 5 includes the results of the extensive computational tests we performed to compare our memetic approach with the mathematical formulations solved by means of the CPLEX solver. Finally, Section 6 presents some final remarks.

2. Problems definitions and relations

2.1. Notation

Let $G = (V, E)$ be a connected undirected input graph, and $T = (V, E')$ be a subgraph of G. Let $V(T)_i \subseteq V$, $i \geqslant 0$ be the set of vertices with degree i in T. Moreover, define $V(T)_B = V \backslash \{V(T)_1 \cup V(T)_2\}$; that is, if T is a spanning tree, $V(T)_B$ is the set of its branch vertices.

Moreover, for each $j \in V$, let $\delta(j, T)$ be the degree of j in T. Note that if $j \in V(T)_i$, then $\delta(j, T) = i$. Finally, for each set of vertices $X \subseteq V$, let $\Delta(X, T)$ be the degree sum of the vertices of X in T $(\Delta(X,T) = \sum_{j \in X} \delta(j,T))$.

In order to better clarify the introduced notation, consider the tree T in Fig. 2. We have that $V(T)_1 = \{a, b, c, d, h\}$, $V(T)_2 = \{g\}$, $V(T)_3 = \{e\}$, $V(T)_4 = \{f\}$, $V(T)_B = \{e, f\}$. Moreover, for example, $\delta(g, T) = 2$, $\Delta(\{a, b, f\}, T) = 6$, $\Delta(V(T)_B, T) = \Delta(\{e, f\}, T) = 7$.

2.2. Definitions

We can now formally define the three problems studied in this paper.

Minimum Branch Vertices Problem (MBV).
Find a spanning tree T of G such that the number of branch vertices is minimized, that is, such that the set $V(T)_B$ has the minimum cardinality.
Minimum Degree Sum Problem (MDS).
Find a spanning tree T *of* G such that the degree sum of the branch vertices is minimized, that is, such that $\Delta(V(T)_B, T)$ is minimized.
Minimum Leaves Problem (ML).
Find a spanning tree T *of* G such that the number of degree-1 vertices is minimized, that is, such that the set $V(T)_1$ has the minimum cardinality.

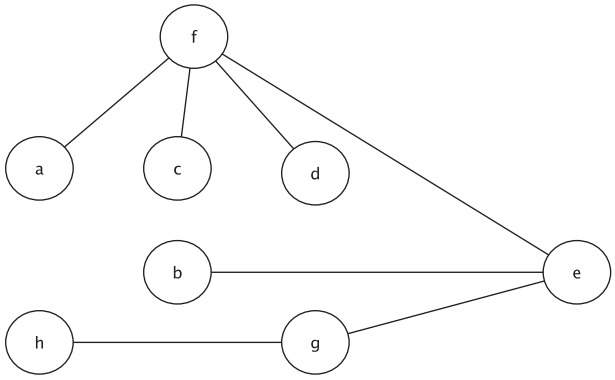


Fig. 2. Example tree T.

2.3. Relations among the problems

In Cerulli et al. (2009) the authors proved that, though being strictly related, MDS and MBV are not equivalent. This was shown by presenting an example graph admitting two distinct optimal solutions for MBV, one of which is also optimal for MDS, while the other is not. However, it was not proved whether an optimal solution for MDS is always optimal for MBV. Here we provide evidence that this is not true, and therefore that optimal solutions for the two problem can be completely distinct.

Consider the example graph in Fig. 3(a). The optimal solution for the MBV problem consists in a spanning tree with a single branch vertex, as shown in Fig. 3(b). However, the optimal solution value of MDS for this graph is six, that is also the degree sum of the two branch vertices of the tree shown in Fig. 3(c), which of course is not an optimal solution for MBV.

It can also be easily shown that the ML problem is not equivalent to neither MBV nor MDS. For example, given the graph in Fig. 4(a), the optimal solution value for MBV is one, and the optimal solution value for MDS is five; Fig. 4(b) shows an optimal tree for both the problems. This solution has five degree-1 vertices, while the optimal ML solution value for this instance is four; the related optimal solution is shown in Fig. 4(c). This solution is not optimal for the other two problems, having two branch vertices with a total degree sum equal to six.

Now, consider the following propositions:

Proposition. *Given a connected undirected graph $G = (V, E)$ and a spanning tree $T = (V, E')$ of G, the following equation holds:*

$$\Delta(V,T) = 2|V| - 2. \tag{1}$$

Proof. Since any spanning tree T of G has exactly $|V| - 1$ edges and any edge of T increases the degree of two nodes by one, if follows that

$$\Delta(V,T) = 2|E'| = 2(|V| - 1) = 2|V| - 2. \tag{2}$$

□

Proposition. *Given a connected undirected graph $G = (V, E)$ and a spanning tree $T = (V, E')$ of G, the following equation holds:*

$$\Delta(V(T)_B, T) = 2|V| - 2 - |V(T)_1| - 2|V(T)_2|. \tag{3}$$

Proof. It is straightforward to observe that

$$\Delta(V(T)_B, T) = \Delta(V,T) - \Delta(V(T)_1, T) - \Delta(V(T)_2, T); \tag{4}$$
$$\Delta(V(T)_1, T) = |V(T)_1|; \tag{5}$$
$$\Delta(V(T)_2, T) = 2|V(T)_2|. \tag{6}$$

By substituting (1), (5) and (6) inside (4), we obtain (3). □

Proposition. *Given a connected undirected graph $G = (V, E)$ and a spanning tree $T = (V, E')$ of G, the following equation holds:*

$$|V(T)_1| = \Delta(V(T)_B, T) - 2|V(T)_B| + 2. \tag{7}$$

Proof. By reformulating (3) we have

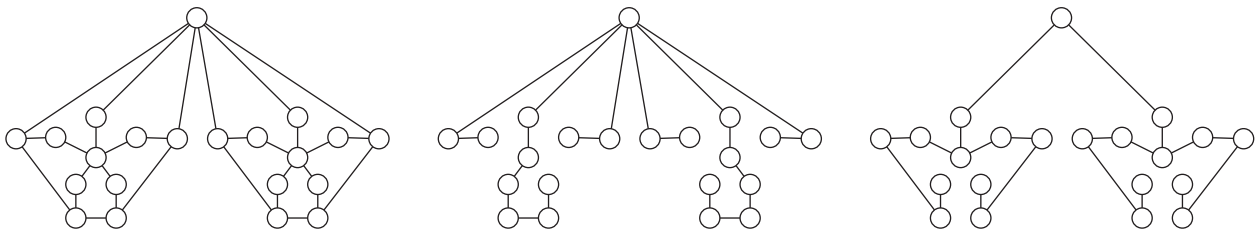
$$|V(T)_1| = 2(|V| - |V(T)_2|) - 2 - \Delta(V(T)_B, T); \tag{8}$$

it is also easy to note that

$$|V| - |V(T)_2| = |V(T)_B| + |V(T)_1|. \tag{9}$$

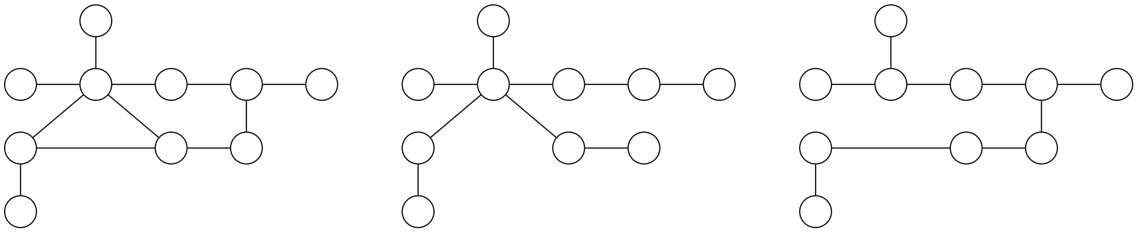
By substituting (9) in (8) and reformulating, we obtain (7). □

We can use (7) to show that ML is a problem of interest in the field of optical network design. As already said in Section 1, many switch devices can only duplicate light signals. Consider a spanning tree $T = (V, E')$ which is used for multicast communications on a graph $G = (V, E)$. Let u be a given vertex which is reached by the signal; no switch devices are needed in u if $\delta(u, T) = 1$ (i.e., u is a leaf) or $\delta(u, T) = 2$ (u just propagates the signal coming from its parent in T to its child). If u is a branch vertex, the signal entering in u must be propagated to its $\delta(u, T) - 1 \geqslant 2$ children; without loss of generality let us call them $v_1, \ldots, v_{\delta(u,T)-1}$. As exemplified in Fig. 1, this can be accomplished using $\delta(u, T) - 2$ devices $(s_1, \ldots, s_{\delta(u,T)-2})$: the signal is sent from u to s_1 and each s_i propagates it to v_i and to either s_{i+1} if $i < \delta(u, T) - 2$ or v_{i+1} if $i = \delta(u, T) - 2$. By iterating this reasoning for each branch vertex, we have that the number of required switch devices is $\Delta(V(T)_B, T) - 2|V(T)_B|$, which is the right hand side of (7) minus a constant factor of 2. Therefore ML can be used to find the spanning tree which requires the minimum number of switch devices.



(a) Example graph (b) Optimal solution for both MBV and MDS (c) Optimal solution for MDS

Fig. 3. MBV and MDS are not equivalent.



(a) Example graph (b) Optimal solution for both MBV and MDS (c) Optimal solution for ML

Fig. 4. MBV, MDS and ML are not equivalent.

2.3.1. Unified objective function

The relations among the three problems can be further highlighted by rewriting their objective functions in terms of a parametric unified objective function, as we will show in the following.

The MBV objective function can be reformulated as

$$\min |V(T)_B| = \min(|V| - |V(T)_1| - |V(T)_2|); \tag{10}$$

since $|V|$ is a constant value,

$$\min |V(T)_B| = -\max(|V(T)_1| + |V(T)_2|) + |V|. \tag{11}$$

Moreover, using (3) the MDS objective function can be reformulated as

$$\min \Delta(V(T)_B, T) = \min(2|V| - 2 - |V(T)_1| - 2|V(T)_2|); \tag{12}$$

since $2|V| - 2$ is a constant value,

$$\min \Delta(V(T)_B, T) = -\max(|V(T)_1| + 2|V(T)_2|) + 2|V| - 2. \tag{13}$$

Finally, it is also easy to note that the ML objective function can be reformulated as

$$\min |V(T)_1| = -\max -|V(T)_1|. \tag{14}$$

We can then use the following objective function to find optimal solutions for the three problems

$$\max(\alpha|V(T)_1| + \beta|V(T)_2|), \tag{15}$$

where

- **for MBV**: $\alpha = 1$, $\beta = 1$;
- **for MDS**: $\alpha = 1$, $\beta = 2$;
- **for ML**: $\alpha = -1$, $\beta = 0$.

3. Mathematical formulations

We present three mathematical formulations for each of the three considered problems. To impose the selection of a spanning tree T of the input graph, two formulations for each of the problems are flow-based, while the remaining three use the well-known Miller–Tucker–Zemlin (MTZ) subtour elimination constraints (see Miller, Tucker, & Zemlin, 1960). The formulations for MBV were originally proposed in Carrabs et al. (in press). Formulations proposed in the literature for problems that impose hard constraints on node degrees, such as the degree-constrained spanning tree problem, cannot be naturally adapted to the problems presented in this work. The section also presents a mixed integer-continuous relaxation for each model. The formulations are defined on directed graphs, therefore we consider a directed version of G that contains both arcs (u, v) and (v, u) for each edge $(u, v) \in E$; let $G^d = (V, E^d)$ be this graph. An arbitrary node $s \in V$ is selected as root node of T.

The models are presented in Sections 3.1–3.3 while relaxations are discussed in Section 3.4.

3.1. MBV formulations

In Sections 3.1–3.3, the three formulations for MBV are presented and analyzed.

3.1.1. MBV Singlecommodity (SC) formulation

$$\min \quad \sum_{v \in V} y_v, \tag{16}$$

$$\text{s.t.} \quad \sum_{(u,v) \in E^d} x_{uv} = 1 \quad \forall v \in V \setminus \{s\}, \tag{17}$$

$$\sum_{(u,v) \in E^d} x_{uv} = n - 1, \tag{18}$$

$$\sum_{(s,v) \in E^d} f_{sv} - \sum_{(v,s) \in E^d} f_{vs} = |V| - 1, \tag{19}$$

$$\sum_{(v,u) \in E^d} f_{vu} - \sum_{(u,v) \in E^d} f_{uv} = -1 \quad \forall v \in V \setminus \{s\}, \tag{20}$$

$$\sum_{(v,u) \in E^d} x_{vu} + \sum_{(u,v) \in E^d} x_{uv} \leqslant \delta(v, G) y_v + 2 \quad \forall v \in V, \tag{21}$$

$$x_{uv} \leqslant f_{uv} \leqslant (|V| - 1) x_{uv} \quad \forall (u, v) \in E^d, \tag{22}$$

$$y_v \in \{0, 1\} \quad \forall v \in V, \tag{23}$$

$$x_{uv} \in \{0, 1\} \quad \forall (u, v) \in E^d, \tag{24}$$

$$f_{uv} \geqslant 0 \quad \forall (u, v) \in E^d. \tag{25}$$

Variables $x_{uv}, f_{uv} \forall (u, v) \in E^d$ determine whether (u, v) is part of T and the amount of flow passing through it, respectively. Each binary variable $y_v \forall v \in V$ is equal to 1 if v is a branch vertex.

The objective function (16) minimizes the number of branch vertices. Constraints (17) make sure that each vertex except the source has exactly one parent in T, while Constraints (18) make sure that exactly $n - 1$ arcs are selected in the solution. Constraints (19) and (20) are flow conservation constraints, imposing that exactly $|V| - 1$ flow units are produced in s and one of them is retained by each node in $V \setminus \{s\}$. Constraints (21) impose vertex v to be a branch if its degree is greater than two in the tree; note that the value of y_v is unconstrained if $\delta(v, T) \leqslant 2$, however in this case it will be set to 0 by the objective function. Finally, Constraints (22) make sure that there is a positive amount of flow only on arcs that belong to T.

3.1.2. MBV Multicommodity (MC) formulation

$$\min \quad \sum_{v\in V} y_v, \tag{26}$$

$$\text{s.t.} \quad (17), (18), (21), (23), (24),$$

$$\sum_{(v,u)\in E^d} f^k_{vu} - \sum_{(u,v)\in E^d} f^k_{uv} = 0 \quad \forall k \in V, v \in V \setminus \{s\}, \ v \neq k, \tag{27}$$

$$\sum_{(s,v)\in E^d} f^k_{sv} - \sum_{(v,s)\in E^d} f^k_{vs} = 1 \quad \forall k \in V \setminus \{s\}, \tag{28}$$

$$\sum_{(k,v)\in E^d} f^k_{kv} - \sum_{(v,k)\in E^d} f^k_{vk} = -1 \quad \forall k \in V \setminus \{s\}, \tag{29}$$

$$f^k_{uv} \leqslant x_{uv} \quad \forall k \in V, \ (u,v) \in E^d, \tag{30}$$

$$f^k_{uv} \geqslant 0 \quad \forall k \in V, \ (u,v) \in E^d. \tag{31}$$

For each $(u, v) \in E^d, k \in V\setminus\{s\}$, flow variable f^k_{uv} is equal to 1 if a unit of flow produced by s and targeted to k passes through (u, v) (that is, if (u, v) is part of the path from s to k in the solution), 0 otherwise. Constraints (27)–(29) are a disaggregated version of the flow conservation constraints, imposing that w produces one unit of flow for each $k \in V \setminus\{s\}$, and that any unit of flow entering into a node v and targeted to k is absorbed if $v = k$, while is forwarded to a different node otherwise. Constraints (30) impose that the units of flow can only pass though arcs that are included into the solution.

3.1.3. MBV Miller–Tucker–Zemlin (MTZ) formulation

$$\min \quad \sum_{v\in V} y_v, \tag{32}$$

$$\text{s.t.} \quad (17), (18), (21), (23), (24),$$

$$t_s = 0, \tag{33}$$

$$t_v \geqslant 1 \quad \forall v \in V \setminus \{s\}, \tag{34}$$

$$(|V| - 2)x_{vu} + |V|x_{uv} + t_u \leqslant t_v + (|V| - 1) \quad \forall (u,v) \in E^d, \tag{35}$$

$$t_v \in \{0, 1, \ldots\} \quad \forall v \in V. \tag{36}$$

Variables t_v contain the values of a labeling function defined on the nodes of G^d. The idea underlying MTZ constraints is to assign to the root s the smallest label (as imposed by Constraints (33) and (34)) and to impose that for each arc (u, v) selected to be part of T vertex u has a smaller label than v. More in detail, Constraints (35) ensure that if $x_{uv} = 1$, then $t_v = t_u + 1$, and are a lifted version proposed in Desrochers and Laporte (1991) of the original MTZ constraints.

3.2. MDS formulations

With respect to the MBV formulations, an additional set of variables z_v is needed in order to take into account the degree of the branch vertices. The models use the constraints of the corresponding MBV formulations to build a spanning tree, as well as Constraints (21) to identify branch vertices. Moreover, Constraints (38) are used by the three models to impose $z_v \geqslant \delta(v, T)$ if $y_v = 1$ and therefore v is a branch vertex, leaving it unconstrained if $y_v = 0$. The objective function (37) minimizes the sum of variables z_v and therefore will impose $z_v = \delta(v, T)$ for the branch vertices, 0 otherwise.

3.2.1. MDS Singlecommodity formulation

$$\min \quad \sum_{v\in V} z_v, \tag{37}$$

$$\text{s.t.} \quad (17)\text{—}(25)$$

$$\sum_{(v,u)\in E^d} x_{vu} + \sum_{(u,v)\in E^d} x_{uv} \leqslant z_v + 2 - 2y_v \quad \forall v \in V, \tag{38}$$

$$z_v \geqslant 0 \quad \forall v \in V. \tag{39}$$

3.2.2. MDS Multicommodity formulation

$$\min \quad \sum_{v\in V} z_v, \tag{40}$$

$$\text{s.t.} \quad (17), (18), (21), (23), (24), (27)\text{—}(31), (38), (39).$$

3.2.3. MDS Miller–Tucker–Zemlin formulation

$$\min \quad \sum_{v\in V} z_v, \tag{41}$$

$$\text{s.t.} \quad (17), (18), (21), (23), (24), (33)\text{—}(36), (38), (39).$$

3.3. ML formulations

In the ML formulations, variables y_v are used to represent whether the vertices are leaves or not in T. Again, constraints to define a tree structure are retained. Constraints (43) ensure that $y_v = 1$ if $\delta(v, T) = 1$. The value of y_v is not constrained if it is not a leaf, however since the objective function (45) minimizes the sum of y_v variables, it will be set to 0.

3.3.1. ML Singlecommodity formulation

$$\min \quad \sum_{v\in V} y_v, \tag{42}$$

$$\text{s.t.} \quad (17)\text{—}(20), (22)\text{—}(25),$$

$$\sum_{(v,u)\in E^d} x_{vu} + \sum_{(u,v)\in E^d} x_{uv} + y_v \geqslant 2 \quad \forall v \in V. \tag{43}$$

3.3.2. ML Multicommodity formulation

$$\min \quad \sum_{v\in V} y_v, \tag{44}$$

$$\text{s.t.} \quad (17), (18), (23), (24), (27)\text{—}(31), (43).$$

3.3.3. ML Miller–Tucker–Zemlin formulation

$$\min \quad \sum_{v\in V} y_v, \tag{45}$$

$$\text{s.t.} \quad \text{Eqs. } (17), (18), (23), (24), (33)\text{—}(36), (38), (39), (43).$$

3.4. Relaxations

For all the presented formulations, we take into account a mixed integer-continuous relaxation, obtained by relaxing integrality on x_{uv} variables.

Furthermore, as already done for example in Akgün and Tansel (2011) and Carrabs et al. (in press), the bounds returned by the relaxations are improved by adding the following set of constraints:

$$x_{uv} + x_{vu} \leqslant 1 \quad \forall (u,v) \in E^d, \quad u < v. \tag{46}$$

4. Memetic algorithm

Memetic algorithms combine population-based metaheuristics (such as genetic or other evolutive algorithms) and more traditional local search schemes. The idea is to obtain a good compromise between the respective strengths of these approaches, in

particular with respect to diversification to explore new regions of the search space, and intensification of the search in promising regions. For a survey on memetic algorithms for discrete optimization, refer to Hao (2012).

Our memetic algorithm (or MA in the following) combines a genetic algorithm with a local search step.

Genetic algorithms (GA) are randomized metaheuristic techniques based on the biological process of natural selection that have been successfully applied to many combinatorial optimization problems. A genetic algorithm emulates the evolutionary process on a *population* composed of solutions (*chromosomes*). While starting populations are often created randomly, new individuals are iteratively formed by recombining together two or more older chromosomes, or by perturbing a single one; the best chromosomes have generally better chances of being selected in these steps. Therefore, new solutions are likely to inherit good characteristics from old solutions, and, by repeating this process over a sufficient number of generations, eventually near-optimal solutions can be reached. For an introduction to genetic algorithms, see for example (Reeves, 2010). Genetic algorithms have been applied with success on Spanning Tree problems; see, for example, (Neumann, 2007; Zhou & Gen, 1999).

The main elements that must be provided in order to implement a GA are:

- A representation scheme for each chromosome. Our algorithm considers only feasible solutions, therefore each chromosome is a spanning tree and is represented internally as a list of its edges.
- A function to evaluate each chromosome (*fitness* function). Our algorithm evaluates the chromosomes using the parametric objective function (15). Depending on the problem on which we intend to focus, the appropriate values for parameters α and β are used.
- Rules to derive new solutions, by combining two parent solutions (*crossover*) and by perturbing a single individual (*mutation*). Our crossover and mutation operators are described in Sections 4.2 and 4.3 respectively.
- Termination criteria for the procedure. In our case, the algorithm ends when a given number of iterations without improvements has been performed.

A high level outline of the procedure is given in Algorithm 1. Each step reported in Algorithm 1 will be explained in detail in the remaining part of this section. The algorithm description makes use of various input parameters, whose setting during our experimental analysis is discussed in Section 5.

Algorithm 1. Memetic Algorithm

1: Build a random population Π
2: **while** iterations without improvements $\leqslant$*max-it* **do**
3: choose two parent chromosomes T_1, T_2
4: perform *crossover* on T_1 and T_2 obtaining T_3
5: perform a *mutation* on T_3 obtaining T_3'
6: perform a *local search* starting from T_3', obtaining T_3''
7: insert T_3'' in Π substituting an older chromosome T_i
8: **end while**

4.1. Parents selection and child insertion policies

As reported in Line 3 of Algorithm 1, at each iteration, two chromosomes are selected for the crossover phase. Our algorithm uses *tournament selection*, which is a scheme commonly used by GAs in order to promote good individuals.

We implemented a simple tournament selection which can be summarized as follows:

- Select randomly h chromosomes from the population Π; let T_1 be the one with the *best* fitness function value among them.
- Select randomly $\frac{h}{2}$ chromosomes from $\Pi\backslash\{T_1\}$; let T_2 be the one with the *best* fitness function value among them.
- Select T_1, T_2 as parents for the crossover phase.

Ties are broken randomly. The idea underlying the implemented tournament scheme is to promote the selection of at least a parent chromosome with a good fitness value, while a higher degree of diversity is left for the selection of the other parent. Of course, the chosen value for parameter h influences both the convergence rate of the algorithm and its capacity to escape from local minima.

The new chromosome T_3'' that will result after the phases reported in Lines 4–6 will replace an older one; therefore the population size $|\Pi|$ is constant throughout every phase of the algorithm. The substituted element is also selected using a tournament mechanism:

- Select randomly k chromosomes from Π; let T_i be the one with the *worst* fitness function value among them.
- Substitute T_i with T_3'' in Π.

4.2. Crossover

The crossover operator builds a new individual from two parents. In classical GAs, crossover is performed by simple recombination of the information contained in the parents. For example, supposing that in a GA chromosomes are represented as lists, two children might be obtained from two parents by swapping between them all the data contained after a given position (*crossover point*). Our crossover, instead, creates new individuals which inherit "good" characteristics from the parents, but are not a direct recombination of them.

More in detail, given the parent chromosomes T_1 and T_2, we start by defining two weight functions w_1 and w_2 on the edges of the input graph $G = (V, E)$; each w_i is based on the characteristics of T_i. Finally, the new spanning tree T_3 is built by finding a Minimum Spanning Tree of the weighted graph $G_w = (V, E, w)$, where

$$w(u, v) = w_1(u, v) + w_2(u, v) \quad \forall (u, v) \in E.$$

The aim of the weight functions is to penalize or promote the selection in T_3 of the edges of E, using three key ideas:

- Strongly promote the selection of chains coming from the parents; this is an obvious choice since we would like to obtain the highest possible number of vertices with degree $\leqslant 2$ in the final solution. The longer is a chain, the more its edges are promoted.
- promote the selection of edges connected to vertices that are branch in the parents; as the procedure converges, some branch vertices are likely to be perceived as required. Edges actually belonging to the parents are more promoted than new edges connected to such vertices.
- Penalize the selection of edges connected to vertices with degree 2 in the chains, in order to avoid new branch vertices.

These guidelines might suggest both to penalize and to promote the same edge. We represent penalizing and promoting weights for each edge (u, v) and parent $T_i = (V, E_i)$ using three positive input

parameters M, Ω, ω, such that $M > 4\Omega$ and $\Omega > 4\omega$. More in detail, w_i is computed as follows:

1. Initialize $w_i(u, v) = 0\ \forall (u, v) \in E$.
2. For each $u \in V(T_i)_B$, add $-\Omega$ to $w_i(u, v)\ \forall (u, v) \in E_i$.
3. For each $u \in V(T_i)_B$, add $-\omega$ to $w_i(u, v)\ \forall (u, v) \in E \backslash E_i$.
4. Remove all edges connected to branch vertices from T_i, obtaining the forest $F_i = (V, E_i')$.
5. For each $(u, v) \in E_i'$, let $l(u, v)$ be the length of the chain it belongs to in F_i, and add $-Ml(u, v)$ to $w_i(u, v)$.
6. For each $u \in V(F_i)_2$, add 2ω to $w_i(u, v)\ \forall (u, v) \in E \setminus E_i'$.

A positive value for $w_i(u, v)$ means that the selection of (u, v) is penalized according to parent T_i, while a negative value means that it is considered a useful edge (recall that the weights are used to compute a minimum spanning tree).

The steps described above are illustrated on the tree shown in Fig. 5. In particular, Fig. 5(b) shows the negative weights applied to promote edges connected to the branch vertex 2. In Fig. 5(c) the negative weights associated with chains are added, and finally Fig. 5(d) includes the penalizing weights for the edges connected to nodes with degree 2 in F_i. It can be noted that each edge with a positive weight would add at least one branch vertex if included in F_i; in particular, the promoting weight assigned to (2,6) for being connected to branch vertex 2 is counterbalanced by its penalization deriving from vertex 6.

The chosen values for parameters ω, Ω and M are reported in Section 5. By imposing $M > 4\Omega$, we make sure that edges belonging to chains are always promoted with respect to other edges (the smallest weight that can be assigned to an edge that is not in a chain is -4Ω, associated to an edge belonging to both parents and connected to two branch vertices in each of them). For a similar reasoning, we also impose $\Omega > 4\omega$.

4.3. Mutation

Given the new individual T_3 resulting from the crossover, the mutation phase operates by applying two different mutation operators M_1, M_2. M_1 is always executed, while M_2 is executed after M_1 with probability p_{m2}. The operators work as follows:

- **M_1**: A leaf node of T_3 is randomly selected and connected to another random node. The resulting cycle is broken randomly obtaining a new spanning tree.
- **M_2**: Randomly eliminates two edges, trying to promote the elimination of branch vertices by repeating the random choice up to a fixed number of iterations, it_{M_2}. The resulting three components are rejoined by randomly selecting two new edges, obtaining the new tree.

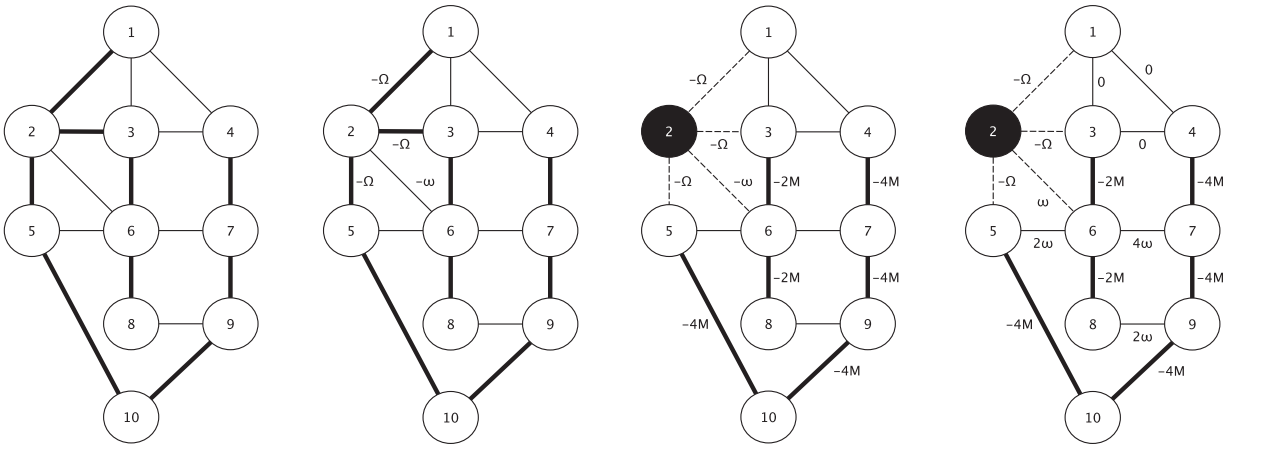
Mutation M_2 is illustrated in Fig. 6, where two branch vertices are removed by selection edges (2,4) and (5,8), and a new branch vertex is created when edges (1,7) and (6,9) (supposing that they exist in G) are chosen to rejoin the three components.

4.4. Local search

At the end of each iteration of the genetic algorithm, a local search is performed on the new chromosome T_3' obtained after the mutation phase. Similarly to the M_1 operator, each iteration of the local search tries to switch one of the edges belonging to the current solution with a new one; however, instead of selecting edges randomly, we look for an *improvement* of the original tree. The improvement is evaluated according to a heuristic function that promotes the introduction of new nodes whose degree is not greater than two, and that strongly penalizes the introduction of new branch vertices. Additional edges added to old branch vertices are also penalized, using values that are inversely proportional to the node degree. More in detail, given a subgraph H of G, we define the following functions:

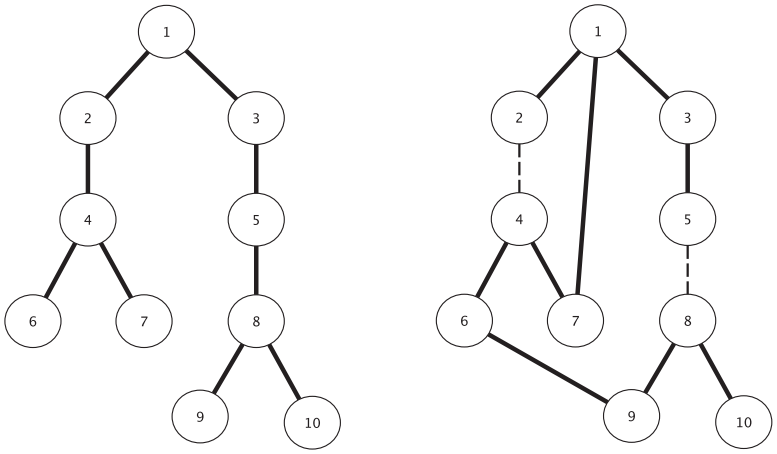
1. Let $im(x, H)$ be a function defined on the nodes of H, such that $im(x, H)$ is equal to: 0 if $\delta(x, H) \leqslant 2$, M if $\delta(x, H) = 3$, $1/\delta(x, H)$ if $\delta(x, H) > 3$.
2. Let $IM((u, v), H)$ be a function defined on the edges of H such that $IM((u, v), H) = im(u, H) + im(v, H)$.

Let (u', v') be an edge of T_3'; moreover, let $(u'', v'') \notin T_3'$ be an edge of G with an endpoint in each of the two connected components that would be created by removing (u', v') from T_3'. We use the IM function to evaluate if replacing (u', v') with (u'', v'') would be a convenient choice as follows:



(a) Example tree T_i (b) Negative weights assigned to edges connected to branch vertex 2 (c) Negative weights assigned to chains (d) Positive weights assigned to edges connected to chains

Fig. 5. Weighting function example for the crossover operation.



(a) Example tree T_i

(b) Mutated tree obtained by substituting $(2,4)$ and $(5,8)$ with $(1,7)$ and $(6,9)$

Fig. 6. Mutation M_2 example.

1. Let $H = T'_3 \cup \{(u'', v'')\}$.
2. If $IM((u'', v''), H) < IM((u', v'), H)$ then $T''_3 = H \setminus \{(u', v')\}$ is an *improvement* of T'_3.

Each iteration of the local search looks for an improvement, by considering all the possible edge substitutions that can be used to obtain a new tree. Each iteration accepts the first improvement found, and the local search phase ends when no improvements are available.

Two local search iterations are shown in Fig. 7. Starting from the tree with two branch vertices in Fig. 7(a), removing (3,4) and rejoining the components using edge (8,9) (supposing that it exists in the graph) produces a new tree that is an improvement of the previous one, as shown in Fig. 7(b). Note that due to the weighting function, the degree of vertex 3 is lowered from 4 to 3, even if the degree of vertex 9 increases from 4 to 5. On the resulting tree, assuming that edge (5,7) also exists in the graph, it allows to decrease again the degree of vertex 3 by removing edge (3,5), as illustrated in Fig. 7(c), producing the tree with a single branch vertex shown in Fig. 7(d).

5. Computational results

In this section, we describe the computational results that we obtained by applying the proposed MA for the three problem variants on a set of test instances. The SC and MTZ mathematical formulations introduced in Section 3 were used to obtain optimal solutions as well as lower bounds for the more complex instances. The MC formulations results are not reported since they did not provide solutions in reasonable computational time for most of the considered test instances.

5.1. Instances description

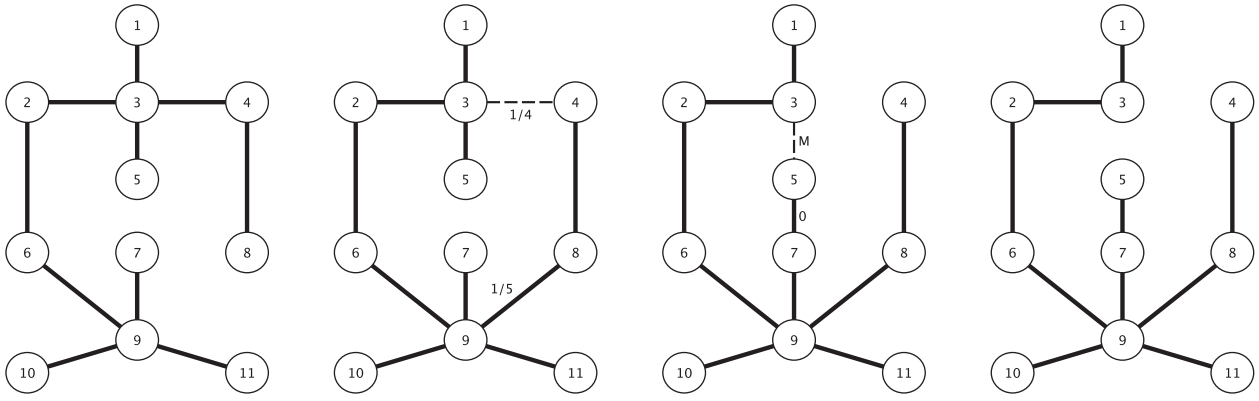
In order to test the performance of our memetic algorithm, we considered a wide set of test instances which belong to two main groups, identified as Type 1 and Type 2 in the following.

The instances belonging to the first group (Type 1), generated according to parameters originally proposed in Carrabs et al. (in press) for the MBV problem, are designed to be sparse in order to require a significant number of branch vertices. We generated instances with 14 different values for the number of nodes $|V|$ between 150 and 1000. The number of edges is generated according to the following formula: $\left\lfloor (|V|-1) + i \times 1.5 \times \left\lceil \sqrt{|V|} \right\rceil \right\rfloor$ with $i = 1, 2, 3, 4, 5$. We randomly generated five instances for each choice of V and i, therefore the total number of Type 1 instances is 350.

Furthermore, in order to test our approaches on more challenging scenarios, we defined instances belonging to Type 2. The instances are generated by merging together short subtours composed of four nodes each. The construction of such instances starts from a first four nodes tour (see Fig. 8(a)). Then, two non-connected nodes of the tour are randomly selected, connected and used to construct a new subtour together with two new nodes, as illustrated in Fig. 8(b). The construction goes on by iteratively adding couple of nodes which are used to produce new tours together with randomly chosen couple of nodes belonging to previous tours. Couple of nodes already belonging to the graph could be randomly chosen more than once, therefore, not every iteration connects two previously disjoint nodes (as in the example in Fig. 8(c)), leading to diversity in terms of nodes degree and overall number of edges. The construction of tours ends when the desired number of nodes $|V|$ is reached; finally, $|V|/2$ random edges are added to the graph to further increase diversity. Instances were generated for 10 different values of $|V|$ between 50 and 1500, generating 10 instances for each choice, for a total of 100 Type 2 instances.

5.2. Parameters and testing environment

Regarding our memetic algorithm, after a tuning phase we chose the following values for the parameters, which seemed to provide a good tradeoff among solution quality and computational time: $|\Pi| = 1000$, *max-it* = 25,000, $h = 8$, $k = 3$, $M = 21$, $\Omega = 5$, $\omega = 1$, $p_{m2} = 0.5$, $it_{m2} = 5$. The MA has been coded in C++. The IBM ILOG CPLEX 12 solver was used to solve the mathematical formulations, considering a time limit of 1 hour for each instance. All tests have been executed on an Intel Xeon 2Ghz workstation with 8 gigabytes of RAM.



(a) Example tree T_i (b) Improved tree obtained by substituting $(3,4)$ with $(8,9)$ (c) Improved tree obtained by substituting $(3,5)$ with $(5,7)$ (d) Resulting tree

Fig. 7. Tree improvements in local search.

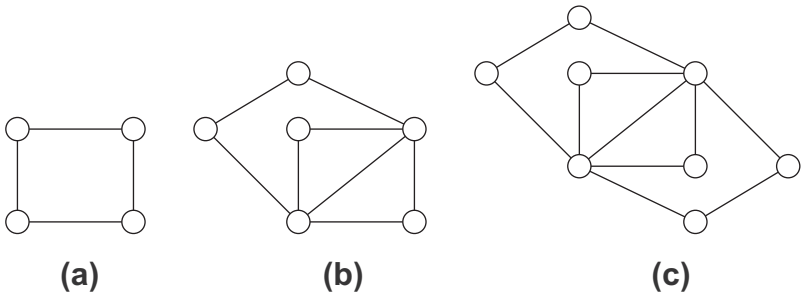


Fig. 8. Type 2 instances construction example.

5.3. Results

Tables 1–3 present comparison between MA and mathematical formulations results on Type 1 datasets, while Tables 4–6 summarize these results for Type 2 datasets. Results are averages of the objective function values and computational times in seconds over all instances with the same value of $|V|$.

For each scenario, comparisons are made with respect to exact results if one among the SC or MTZ models allow to obtain more than half of the optimal solutions within the considered time limit (that is, if at least 13 out of 25 exact solutions were obtained for Type 1 instances, or 6 out of 10 for Type 2). Otherwise, the lower bounds given by the mixed relaxations are used for the comparisons. The SC relaxation provided better or equal lower bounds with respect to the MTZ one every instance (see the discussion in Section 5.3.1); therefore, the mixed relaxation of SC is used, unless it fails to provide more than half of the solutions of a given scenario within the 1-hour time limit, in which case the MTZ relaxation is used instead. The specific formulation used for the comparison on each scenario is reported under the *type* column in Tables 1–6, where EX and MR stand for *exact* and *mixed relaxation*, respectively. The number of actual instances solved by the considered model for each scenario is reported in the *# inst* column. When this number is lower than the size of the scenario, averages are only evaluated on the instances that were solved by the model.

We now analyze the results on Type 1 instances, as reported in Tables 1–3. On such instances, the MTZ formulation is overall able to find more solutions than the SC one and converges in faster times, therefore it was used for our comparisons on exact solutions. In particular, it is able to find 311 out of 325 optimal solutions on the instances with up to 750 nodes for MBV, 293 out of 300 optimal solutions on the instances with no more than 500 nodes for MDS, and all the 350 optimal solutions for ML. It is then possible to accurately verify the performances of the memetic algorithm on these instances. It can be noted that the MA returns very high quality solutions for all the three problems. More in detail, for MBV, on instances up to $|V| = 500$ the gap among the average solutions returned by the MA and the optimal ones (that have on average up to 108 branch vertices) is never higher than 2 branch

Table 1
Genetic/models comparison for MBV on Type 1 instances.

Instances	Memetic		Model			
$\|V\|$	Sol	Time	Sol	Time	# Inst	Type
150	23.28	5.29	22.80	3.74	25	EX-MTZ
160	25.68	6.04	25.04	41.53	25	
170	27.25	6.99	26.67	22.51	24	
180	29.96	7.49	29.08	8.60	25	
190	32.60	8.43	32.12	79.13	25	
200	34.43	9.04	33.61	4.48	23	
250	45.64	16.09	44.60	13.16	25	
300	58.56	22.67	57.36	14.15	25	
350	70.65	33.73	69.35	123.57	23	
400	83.32	47.27	81.84	105.92	25	
450	96.39	55.26	94.74	188.58	23	
500	109.96	67.84	108.00	285.76	24	
750	207.21	130.37	203.11	680.47	19	
1000	278.23	275.73	272.32	243.11	23	MR-SC

Table 2
Genetic/models comparison for MDS on Type 1 instances.

Instances	Memetic		Model			
$\|V\|$	Sol	Time	Sol	Time	# Inst	Type
150	98.04	5.99	96.92	1.77	25	EX-MTZ
160	107.44	6.74	106.52	3.29	25	
170	115.46	7.46	114.38	6.14	24	
180	125.64	8.50	124.56	10.12	25	
190	136.64	9.51	135.60	16.45	25	
200	141.60	10.54	140.32	19.22	25	
250	191.64	16.56	190.12	31.70	25	
300	247.21	23.54	245.21	45.86	24	
350	289.43	34.25	286.78	28.10	23	
400	346.60	48.35	343.96	21.54	25	
450	398.83	63.48	395.83	47.46	23	
500	450.33	73.02	446.71	298.56	24	
750	844.00	156.46	835.00	675.76	16	MR-SC
1000	1154.48	297.29	1142.57	369.17	24	MR-MTZ

Table 3
Genetic/models comparison for ML on Type 1 instances.

Instances	Memetic		Model			
$\|V\|$	Sol	Time	Sol	Time	# Inst	Type
150	50.52	5.23	50.40	0.14	25	EX-MTZ
160	55.40	5.78	55.12	0.09	25	
170	58.52	6.73	58.20	0.17	25	
180	64.92	7.77	64.60	0.22	25	
190	70.72	8.08	70.36	0.90	25	
200	73.08	9.38	72.80	0.12	25	
250	98.16	14.74	97.68	0.18	25	
300	124.40	24.85	123.36	0.31	25	
350	147.12	28.83	146.00	0.25	25	
400	175.72	36.43	174.52	0.32	25	
450	198.76	54.02	197.64	0.48	25	
500	228.24	60.47	226.44	0.33	25	
750	437.00	99.10	437.00	0.38	25	
1000	595.00	180.40	595.00	0.71	25	

Table 4
Genetic/models comparison for MBV on Type 2 instances.

Instances	Memetic		Model			
$\|V\|$	Sol	Time	Sol	Time	# Inst	Type
50	1.60	0.93	1.60	0.18	10	EX-SC
100	3.60	2.91	3.50	1.01	10	
200	6.80	10.79	6.30	3.75	10	
300	10.20	26.99	8.90	42.96	10	
400	14.10	45.58	11.70	111.18	10	
500	18.63	77.69	15.88	461.56	8	
750	27.13	156.67	22.13	1047.16	8	
1000	39.00	371.63	32.20	526.39	10	MR-SC
1250	50.22	536.68	41.56	454.50	9	
1500	57.20	1001.68	47.20	987.00	10	

vertices. With $|V| = 750$, the MA averages solutions that are 2.01% higher than the average optimum, while on instances with $|V| = 1000$, where the comparison is obtained with the lower bounds given by MR-SC, this percentage gap is equal to 2.17%.

The percentage gap is always consistently below 1.16% for the MDS problem, even for scenarios when relaxed solutions have to be taken into account, and below 0.85% for the ML problem.

Regarding computational times, it can be noted that they are reasonable for the MA procedure regardless of the considered problem, averaging under 6 seconds for the smaller instances, under 300 seconds for the bigger ones for MBV and MDS, and under

Table 5
Genetic/models comparison for MDS on Type 2 instances.

Instances	Memetic		Model			
$\|V\|$	Sol	Time	Sol	Time	# Inst	Type
50	8.80	0.98	8.70	0.42	10	EX-SC
100	22.90	3.06	22.50	3.35	10	
200	45.30	12.25	43.10	64.47	10	
300	70.90	31.14	67.70	134.50	10	
400	95.00	54.57	89.44	724.32	9	
500	134.17	84.70	126.50	438.35	6	
750	192.70	247.22	171.48	486.82	10	MR-SC
1000	293.00	481.89	260.20	1569.58	6	
1250	355.80	743.00	316.90	14.37	10	MR-MTZ
1500	409.00	1342.58	361.10	23.82	10	

Table 6
Genetic/Models comparison for ML on Type 2 instances.

Instances	Memetic		Model			
$\|V\|$	Sol	Time	Sol	Time	# Inst	Type
50	7.40	0.97	7.40	0.15	10	EX-MTZ
100	17.30	2.86	17.30	0.75	10	
200	32.40	9.70	32.20	2.51	10	
300	51.50	20.66	51.30	7.26	10	
400	66.30	38.95	65.70	21.19	10	
500	88.50	62.21	88.00	24.01	10	
750	136.00	153.02	135.20	124.00	10	
1000	196.50	438.70	194.75	466.89	8	
1250	256.17	615.09	252.83	826.64	6	
1500	284.90	1143.89	274.20	2077.56	10	MR-SC

200 seconds for the ML problem. However, while the mathematical models for the MBV and MDS problem become more burdensome as the size of the problem increases, and cannot be used in reasonable time for the highest values of $|V|$ as shown in Tables 1 and 2, ML could be solved very efficiently using the proposed models, therefore our proposed memetic approach results to be unnecessary for this problem on Type 1 instances. This motivated our search for more challenging test scenarios, which led to the definition of Type 2 instances.

As shown in Tables 4–6, the MA reported good solutions on all scenarios where comparisons with exact solutions could be made. The EX-SC model was used for the first 7 scenarios for MBV and for the first 6 scenarios for MDS, since for several datasets EX-MTZ returned optimal solutions for less instances (see the discussion in Section 5.3.1). The maximum average gap among exact and heuristic solutions is 5 for MBV, 7.67 for MDS and 3.34 for ML (for scenarios where optimal solutions average to 22.13, 126.50 and 252.83, respectively). Solution quality is particularly high for ML, whose percentage gap is equal to 1.32% for $|V| = 1250$ and below 1% for smaller instances.

In terms of running time, the MA averages under 1 seconds for the smaller instances, and up to 1001.68 seconds for MBV, 1342.58 seconds for MDS and 1143.89 seconds for ML on instances with $|V| = 1500$. For instances with up to 1000 nodes, the average running time is never higher than 500 seconds. The mathematical models become ineffective to find optimal solutions on scenarios with $|V| = 1000$ for MBV, $|V| = 750$ for MDS and $|V| = 1500$ for ML.

5.3.1. Mathematical formulations comparison

In Tables 7–9 we compare the performances of the MTZ and Single-Commodity formulations on some sample scenarios belonging to Type 2. More in detail, we report the results for the instances with $|V| = 400$ and 500 for MBV and MDS, and of the ones with $|V| = 500$ and 700 for ML. The results on these scenarios are largely

Table 7
MBV models comparison.

Instances	EX		MR			
$\|V\|$	MTZ time	SC time	MTZ time	SC time	MTZ sol	SC sol
400	6.57	30.79	0.45	7.61	11.00	11.00
	TL	35.91	0.61	7.39	10.00	11.00
	2.80	16.15	0.42	7.72	14.00	15.00
	8.42	49.52	0.97	10.87	11.00	11.00
	19.48	287.66	1.16	16.87	11.00	11.00
	7.77	24.02	0.60	9.25	11.00	12.00
	3.01	49.54	0.31	6.61	15.00	16.00
	7.20	61.40	0.61	13.39	9.00	9.00
	2019.27	262.93	1.29	14.83	10.00	10.00
	55.14	293.86	2.10	35.44	11.00	11.00
500	TL	2588.00	6.52	149.39	15.00	16.00
	TL	375.16	0.77	15.06	14.00	16.00
	TL	TL	1.52	65.80	15.00	15.00
	47.21	141.47	0.90	14.38	19.00	20.00
	5.57	93.08	0.85	12.91	16.00	16.00
	12.55	236.37	0.81	8.44	14.00	14.00
	3.20	61.47	0.76	22.16	16.00	16.00
	3.43	42.16	0.84	11.17	16.00	16.00
	13.78	154.74	1.02	22.11	13.00	13.00
	TL	TL	2.06	397.23	13.00	13.00

Table 8
MDS models comparison.

Instances	EX		MR			
$\|V\|$	MTZ time	SC time	MTZ time	SC time	MTZ sol	SC sol
400	TL	409.71	1.05	11.96	80.00	80.01
	TL	162.35	0.78	18.08	81.00	83.01
	2637.52	1031.34	0.75	8.54	77.50	79.51
	50.83	136.32	1.02	32.64	79.00	79.04
	270.22	3269.10	2.25	152.45	86.00	86.05
	61.98	37.81	0.85	10.77	95.00	97.02
	72.27	163.65	0.76	14.66	122.00	124.02
	510.03	278.05	1.11	37.79	75.00	75.05
	TL	TL	7.96	217.97	74.00	74.10
	3081.47	1030.58	6.51	157.21	77.00	77.03
500	TL	TL	24.49	388.49	120.00	120.01
	TL	723.12	1.32	51.70	96.00	100.08
	TL	TL	16.18	657.21	99.00	99.02
	72.33	84.30	1.11	14.19	125.00	127.04
	TL	1285.86	1.18	133.28	118.00	118.04
	111.58	168.69	2.33	14.14	123.00	123.02
	59.44	314.38	0.95	150.06	151.00	151.01
	111.52	53.72	1.35	17.31	112.00	112.03
	2841.39	TL	6.01	17.78	92.50	92.52
	TL	TL	11.97	281.21	99.00	99.03

Table 9
ML models comparison.

Instances	EX		MR			
$\|V\|$	MTZ time	SC time	MTZ time	SC time	MTZ sol	SC sol
500	3.91	8.20	0.80	10.81	83.00	91.00
	13.90	18.27	1.16	20.92	66.00	76.00
	14.99	68.88	1.34	4.52	70.00	73.00
	19.47	7.86	0.37	7.03	87.00	93.00
	29.23	15.99	0.47	14.04	86.00	89.00
	45.50	15.93	0.82	8.79	96.00	98.00
	12.10	6.49	0.44	25.19	120.00	122.00
	23.53	42.76	0.51	30.90	81.00	83.00
	28.49	24.75	0.70	6.99	66.00	71.00
	49.02	103.98	0.54	13.67	72.00	76.00
750	652.66	246.11	1.36	89.45	105.00	117.00
	63.77	116.53	1.51	40.28	113.00	122.00
	72.75	137.10	1.83	23.21	135.00	140.00
	24.13	220.95	1.12	70.31	118.00	120.00
	62.68	40.90	1.91	25.41	151.00	153.00
	77.28	52.39	1.02	79.91	138.00	145.00
	42.41	53.79	1.04	78.99	136.00	143.00
	28.91	54.70	1.24	15.37	141.00	143.00
	123.48	141.92	1.12	373.97	121.00	122.00
	91.91	204.26	1.53	68.61	114.00	118.00

representative of the behavior of the models on the whole Type 2 dataset.

For each problem and each instance, we compare the running time required to solve the MTZ and SC mathematical models, as well as their mixed relaxations (EX and MR columns, respectively). For the mixed relaxations, we also compare the quality of the returned lower bounds. A TL value means that the 1-hour time limit was reached.

It can be seen that the MTZ formulation returns fewer optimal solutions than SC for MBV and MDS, which led to the use of this second formulation for the comparisons in Tables 4 and 5.

For the ML problem, both formulations are able to find all the solutions within the time limit, with MTZ overall converging slightly faster.

Regarding the mixed relaxations, it can be noted that, as previously said, for all the problems the SC formulation returns equal or better bounds, at the expense of longer computational times.

6. Conclusions

In this paper, we show that finding a spanning tree with the minimum number of leaves is a relevant problem in the context of multicast communications on optical networks. In particular, we demonstrate that it can be used to model the problem of minimizing the number of required light-splitting devices more accurately than two problems already proposed in the literature to solve the same issue (namely, the Minimum Branch Vertices and the Minimum Degree Sum Problems). Moreover, we propose a unified memetic procedure that makes use of a common set of rules to produce accurate solutions for the three problems, as shown by our computational tests.

Future research will be aimed at obtaining a better characterization of the similarities among the three problems, looking for possible dominance relations under particular assumptions. Fur-

ther efforts will be also spent at producing improved resolution approaches and test instances coming from real-world applications.

References

Gargano, L., Hell, P., Stacho, L., & Vaccaro, U. (2002). Spanning trees with bounded number of branch vertices. In *Automata, languages and programming. Lecture notes in computer science* (Vol. 2380, pp. 355–365). Berlin/Heidelberg: Springer.

Carrabs, F., Cerulli, R., Gaudioso, M., & Gentili, M. (in press). Lower and upper bounds for the spanning tree with minimum branch vertices. *Computational Optimization and Applications*. http://dx.doi.org/10.1007/s10589-013-9556-5 (in press).

Gargano, L., & Hammar, M. (2003). There are spanning spiders in dense graphs (and we know how to find them). In *Automata, languages and programming. Lecture notes in computer science* (Vol. 2719, pp. 802–816). Berlin/Heidelberg: Springer.

Gargano, L., Hammar, M., Hell, P., Stacho, L., & Vaccaro, U. (2004). Spanning spiders and light-splitting switches. *Discrete mathematics, 285*(1), 83–95.

Ribeiro, C. C., & de Souza, M. C. (2002). Variable neighborhood search for the degree-constrained minimum spanning tree problem. *Discrete Applied Mathematics, 118*(1–2), 43–54.

de Souza, M. C., & Martins, P. (2008). Skewed VNS enclosing second order algorithm for the degree constrained minimum spanning tree problem. *European Journal of Operational Research, 191*(3), 677–690.

Caccetta, L., & Hill, S. (2001). A branch and cut method for the degree-constrained minimum spanning tree problem. *Networks, 37*(2), 74–83.

Duhamel, C., Gouveia, L., Moura, P., & Souza, M. D. (2011). Models and heuristics for the k-degree constrained minimum spanning tree problem with node-degree costs. *Networks, 60*(1), 1–18.

Cerulli, R., Gentili, M., & Iossa, A. (2009). Bounded-degree spanning tree problems: Models and new algorithms. *Computational Optimization and Applications, 42*(3), 353–370.

Sundar, S., Singh, A., & Rossi, A. (2012). New heuristics for two bounded-degree spanning tree problems. *Information Sciences, 195*, 226–240.

Lu, H. -I., & Ravi, R. (1996). *The power of local optimization: Approximation algorithms for maximum-leaf spanning tree*. Tech. Rep. CS-96-05, Brown University, RI (January).

Salamon, G., & Wiener, G. (2008). On finding spanning trees with few leaves. *Information Processing Letters, 105*(5), 164–169.

Fernandes, L., & Gouveia, L. (1998). Minimal spanning trees with a constraint on the number of leaves. *European Journal of Operational Research, 104*(1), 250–261.

Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer programming formulations and travelling salesman problems. *Journal of the ACM, 7*(4), 326–329.

Desrochers, M., & Laporte, G. (1991). Improvements and extensions to the Miller–Tucker–Zemlin subtour elimination constraints. *Operations Research Letters, 10*(1), 27–36.

Akgün, I., & Tansel, B. Ç. (2011). New formulations of the hop-constrained minimum spanning tree problem via Miller–Tucker–Zemlin constraints. *European Journal of Operational Research, 212*(2), 263–276.

Hao, J.-K. (2012). Memetic algorithms in discrete optimization. In *Handbook of memetic algorithms. Studies in computational intelligence* (Vol. 379, pp. 73–94). Berlin/Heidelberg: Springer.

Reeves, C. R. (2010). Genetic algorithms. In *Handbook of metaheuristics. International series in operations research and management science* (Vol. 146, pp. 109–139). US: Springer.

Zhou, G., & Gen, M. (1999). Genetic algorithm approach on multi-criteria minimum spanning tree problem. *European Journal of Operational Research, 114*(1), 141–152.

Neumann, F. (2007). Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. *European Journal of Operational Research, 181*(3), 1620–1629.