Comput Optim Appl
DOI 10.1007/s10589-017-9924-7

Decomposition methods based on articulation vertices for degree-dependent spanning tree problems

Mercedes Landete[1] · **Alfredo Marín**[2] · **José Luis Sainz-Pardo**[1]

Received: 28 September 2016

Abstract Decomposition methods for optimal spanning trees on graphs are explored in this work. The attention is focused on optimization problems where the objective function depends only on the degrees of the nodes of the tree. In particular, we deal with the Minimum Leaves problem, the Minimum Branch Vertices problem and the Minimum Degree Sum problem. The decomposition is carried out by identifying the articulation vertices of the graph and then its blocks, solving certain subproblems on the blocks and then bringing together the optimal sub-solutions following adequate procedures. Computational results obtained using similar Integer Programming formulations for both the original and the decomposed problems show the advantage of the proposed methods on decomposable graphs.

Keywords Integer Programming · Spanning tree · Articulation vertex · Block-cutpoint graph

1 Introduction

In an undirected connected graph with n nodes, a spanning tree (ST) is defined by a connected subgraph with n nodes and $n - 1$ edges. The degree of a node is the number of its incident edges. Apart from the well-known minimum ST problem (STP), many special cases of optimum STs in a graph or in a network have been taken into consideration in the literature due to both their theoretical and practical interest. Some

✉ Mercedes Landete
landete@umh.es

[1] Universidad Miguel Hernandez de Elche, Elche, Spain

[2] Universidad de Murcia, Murcia, Spain

Published online: 13 July 2017

of them are the degree-constrained ST [17], the ST of Hubs [6], edge-equitable STs [11], the full degree STP [1] and the minimum-degree STP [9].

Inside the family of ST optimization problems defined on graphs without weights on the edges, i.e., where the objective depends only on the structure of the tree, we focus on three closely related but independent NP-hard problems: The Minimum Leaf STP (from now on ML), that aims to find the ST with the minimum number of leaves, the Minimum Degree Sum of Branch Vertices STP (from now on MDS), whose objective function is the sum of the degrees of all the vertices except those with degrees 1 and 2, and the Minimum Branch Vertices STP (from now on MBV), that looks for the ST with minimum number of vertices with degree greater than two. The common thread that links these three problems is an objective function solely calculated from the degrees of the nodes in the ST. Note that the last two problems, MDS and MBV, take into consideration nodes with degree greater than two, the so-called *branching nodes* or simply *branches*.

The MBV was proposed in Gargano et al. [10], in the context of the design of optical networks: A light signal must be split by a *switch* in nodes of degree greater than two, and the cost of the network depends on the number of switches. They showed the problem to be NP-hard. Cerulli et al. [4] proposed an Integer Programming formulation for the MBV, designed three different heuristic strategies, namely, the edge weighting strategy, the node coloring strategy and a combined strategy. They also generated a set of instances, comparing the heuristic solutions with the optima reported by a solver running the IP formulation. Silva et al. [21] proposed an adaptation of the Iterative Refinement approach to solve the MBV. Sundar et al. [25] designed two heuristic algorithms, one that built trees by interchanging edges and another one in the family of ant colony heuristics. Merabet et al. [16] considered the IP formulation developed in [4] to deal with a variant of the problem with application in optical networks which imposes branches to belong to a subset of nodes. Carrabs et al. [2] considered four IP formulations to generate bounds by means of three different relaxations, a Lagrangian relaxation, a continuous relaxation and a mixed integer-continuous one, and then used a single commodity flow formulation to optimally solve some instances. Silva et al. [22] checked another edge-swap heuristic on a new set of instances. The three smallest IP formulations were again studied in Cerrone et al. [3], and a memetic heuristic algorithm was proposed by these authors, using old and new instances in their computational study. Matsuda et al. [14] gave bounds on the number of branch vertices in a certain class of graphs called claw-free graphs. Öncan [18] proposed valid inequalities to improve one of the Integer Programming formulations available for the problem. Rossi et al. [19] designed and implemented a branch-and-cut algorithm for the MBV. Marín [13] presented a branch-and-cut algorithm based on an enforced Integer Programming formulation, which could solve many more instances than previous methods. For very large instances that could not be solved exactly, a heuristic two-stage method that gave very good approximate solutions was also developed. Silvestri *et al.* [23] modeled the MBV by means of two Integer Programming formulations; one of them, the so-called hybrid formulation, used variables associated to edges and arcs. They also derived valid inequalities and proved that some of them define facets. Melo et al. [15] considered a combinatorial lower bound for the MBV, from which they proposed a decomposition approach, breaking down the problem into smaller

subproblems, and proposed constructive heuristics which take into consideration the structure of the problem. Finally, Chimani and Spoerhase [5] looked for algorithms for MBV with provable quality guarantees.

Regarding the ML, complexity results were presented in Lu and Ravi [12], and some related problems were studied in Salamon and Wiener [20] and Fernandes and Gouveia [8]. A memetic heuristic algorithm for the problem was proposed by Cerrone et al. [3].

Finally, the MDS was proposed and studied in Cerulli et al. [4], giving mathematical formulations and heuristic procedures for it. New heuristics for the MDS were developed in Sundar et al. [25], and a memetic heuristic algorithm was proposed by Cerrone et al. [3]. In this last paper, the independence between the three problems was proved.

Although there has been a great interest in these problems in the recent past, especially in the MBV, most of the papers have been limited to the development of approximate (heuristic) solution methods. Even among the papers where the exact resolution of the problems was approached, the focus has been on the construction of Integer Programming formulations to be directly implemented on a commercial solver. To the best of our knowledge, only a couple of papers have exploited the structure of the underlying graph. First, Marín [13] proposed to add some special edges to the optimal tree from the beginning, and this idea together with other improvements in the formulation of the MBV was the key to significant time reductions in the solution procedure. Very recently, Melo et al. [15] observed that when the removal of a node (and associated edges) decomposes the graph into at least three connected components, this node will be a branch at any solution of the MBV. Then they solved independently the MBV on each connected component and linked the optimal trees to produce the optimal solution to the MBV in the original graph, thus achieving important time reductions. When the removal of the node decomposes the graph into only two components, the node can be or not a branch in the final solution. Since this decomposition does not lead to any evident conclusion about what to do with the two subgraphs thus obtained, this case has not been taken into account until now in the literature. Moreover, the other two problems, ML and MDS, have not been approached from the decomposition point of view.

This article is aimed to bridge this gap, studying the resolution of the three problems, ML, MBV and MDS, when the graph contains these special nodes whose removal produces two or more connected components. Ad-hoc algorithms for each of the three problems are developed that bring together partial solutions to the corresponding problems on the components produced by the removal of the nodes, guaranteeing the optimality of the global solution. In Sect. 2, some basic concepts on graph theory are refreshed and other new necessary definitions are introduced. In Sect. 3, the decomposition of each of the three problems under consideration is analyzed and the main results and algorithms are presented. Section 4 is devoted to the adaptation of several formulations to make them able to solve the new subproblems of interest required by the algorithms. Computational results are presented in Sect. 5 that show the efficiency of the methods when the graph can be decomposed on large enough subgraphs, and some conclusions close the paper.

2 Basic concepts

Some basic concepts on Graph Theory are introduced below. The notation corresponds to that of Diestel's book [7] to which the interested reader is referred for further definitions and properties.

Let $G = (V, E)$ be a simple graph. Two vertices $u, v \in V$ are *adjacent*, or *neighbors*, if $uv \in E$, and then uv *incides* on u and v. The *degree* of a vertex $v \in V$ is the number of its neighbors and it is denoted by $\delta(v, G)$. The vertex set of a graph G is referred to as $V(G)$ and its edge set is denoted by $E(G)$.

The subgraph of G *induced* by $V' \subseteq V$ is the graph $G_{V'} = (V', E')$ where E' is the set of all the edges $uv \in E$ with $u, v \in V'$. The *union* of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the graph $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. If $U \subset V$, then $G - U$ is the graph which is obtained from G by deleting all the vertices in U and their incident edges.

A *path* is a non-empty graph $P = (V, E)$ of the form $V = \{v_0, v_1, \ldots, v_k\}$, $E = \{v_0v_1, v_1v_2, \ldots, v_{k-1}v_k\}$ where the v_i are all distinct. The vertices v_0 and v_k are *linked* by P and are called its *ends*; the vertices $v_1, \ldots, v_{k-1}$ are the *inner* vertices of P. A cycle is a graph with at least three vertices that is obtained adding to a path an edge incident on both ends. A *bridge* in G is an edge that does not lie on any subgraph of G that defines a cycle. A graph is called *connected* if there exists a path linking each pair of vertices. In this paper, it is assumed that the graphs are connected.

A graph is called *bipartite* if V admits a partition into two classes such that every edge incides in both classes. G is called *k-connected* (for $k \in \mathbb{N}$) if $|V| > k$ and $G - U$ is connected for every set $U \subseteq V$ with $|U| < k$. Note that connection and 1-connection are equivalent concepts. A vertex $c \in V$ is an *articulation vertex* of G (also called *cut vertex*) if $G_{V\setminus\{c\}}$ is disconnected.

A *block* of G is a maximal 2-connected subgraph of G. Different blocks of G overlap in at most one articulation vertex and G is the union of its blocks. The *block-cutpoint* graph of G, denoted by BC(G), is the bipartite graph with vertex set $C \cup B$ where C is the set of articulation vertices of G, B is the set of blocks of G and nodes $c \in C$ and $b \in B$ are neighbors iff c is a node of block b. Figure 1 shows a graph G and its block-cutpoint graph. G has 13 blocks and the set of articulation vertices is $C = \{5, 6, 9, 14, 18, 22, 23, 25\}$. Blocks are induced subgraphs of G. For instance, block B1 is the subgraph of G induced by nodes 1 and 5 of G and B5 is the subgraph of G induced by nodes 5, 6, 7, 8 and 9.

A graph is a *tree* if any two vertices are ends of a unique path (alternatively G is a tree if it is connected and $|E(G)| = |V(G)| - 1$). A vertex of a tree is called a *leaf* if it has degree equal to one and it is called a *branch (vertex)* if it has degree greater than two. Abusing the notation, a node of degree one on any graph (not necessarily a tree) will be also called a leaf of the graph, and the unique node in a graph with $n = 1$ will also be called a leaf. Sometimes one vertex of a tree is considered as special and is called the *root*. Choosing a root in a tree imposes a partial ordering on V by letting $u \prec v$ iff u is an inner vertex of the unique path from the root to v. A subgraph (V', E') of (V, E) is a *spanning tree* (ST) of G if it is a tree and $V' = V$. It is well known that every connected graph contains a spanning tree and also that the block-cutpoint graph of a connected graph is always a tree. The blocks which are leaves of BC (G)

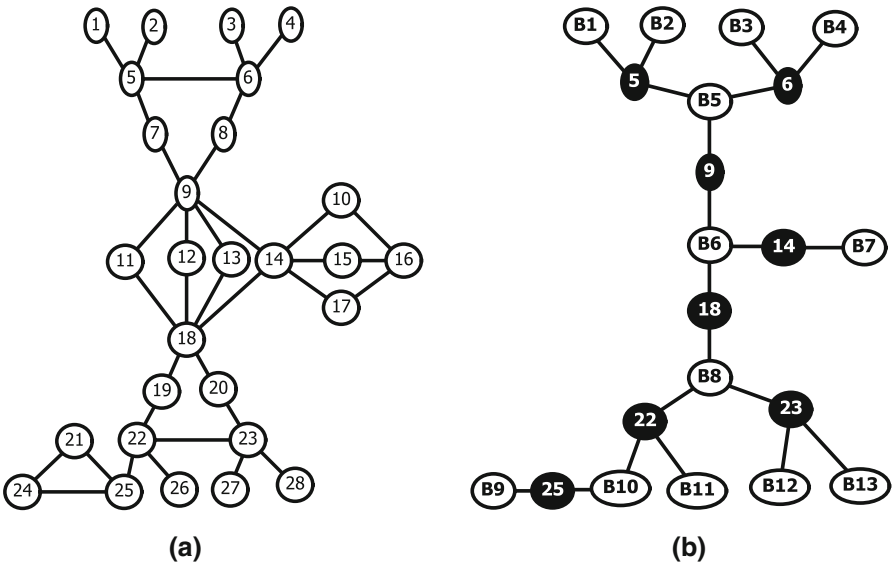

Fig. 1 A graph with 28 nodes and its block-cutpoint graph

are called *leaf blocks* of G. Note that all leaves of BC (G) are blocks. In the example of Fig. 1b, BC (G) has 9 leaf blocks, B1, B2, B3, B4, B7, B9, B11, B12 and B13.

When needed, the optimization problems we are interested in (ML, MBV and MDS, respectively) will be denoted with a specification of the graph on which they are going to be solved (ML(G), MBV(G) and MDS(G), resp.). The optimal value of an optimization problem $(\cdot)$ will be denoted by $v^*(\cdot)$.

It is obvious that the union of one spanning tree of each of the blocks of G is an ST of G. However it does not hold that solving MDS(B) (respectively MBV(B)) for each block B of G and then gathering the corresponding optimal STs, the resulting tree be an optimal solution to MDS(G) (resp. MBV(G)). How to obtain optimal spanning trees for ML(G), MBV(G) and MDS(G) as the union of (not necessarily optimal) spanning trees of their blocks is the matter of the analysis developed in this paper. To this end, some additional non-standard notation is needed.

The letter a will denote the number of blocks of G. Let $B_1 = (V_1, E_1), \ldots, B_a = (V_a, E_a)$ be the blocks of G. Let $C \subset V$ be the set of articulation vertices of G and let $C_i = C \cap V_i$ be the set of articulation vertices of G contained in block i $\forall i \in \{1, \ldots, a\}$ (that is to say, the neighbors of B_i in BC(G)).

Note that an articulation vertex c with degree two in BC(G) can be or not a branch vertex of a given ST. Since the degree of c is the sum of its degrees in the subtrees associated to the blocks c belongs to, an auxiliary function $\gamma : C \to \mathbb{N}_+^*$ which assigns a nonnegative integer to each articulation vertex will be defined to represent the partial incidence degree of c when only part of the graph is considered. From this partial information and given an ST T_i of a block B_i, we say that $v \in C_i$ is a γ-*branch vertex* of T_i if $\delta(v, T_i) \geq 3 - \gamma(v)$. If $\gamma(v) \geq 2$, the inequality is trivially satisfied. If $\gamma(v) = 0$, v is a γ-branch vertex of T_i iff it is a branch vertex of T_i. If $\gamma(v) = 1$, v is a γ-branch vertex of T_i iff it is not a leaf of T_i.

Determining which optimization ST problems must be solved at each block of BC(G) in order to, bringing all the optimal STs together, obtain an optimal solution to the optimization problem on the original graph G, is the goal of the forthcoming section. The following result states that if an articulation vertex splits a graph G in a group of blocks, any ST of G has at least one edge in each block which incides in the articulation vertex, and will be used when needed.

Remark 1 Let c be an articulation vertex of G and let T be an ST of G. For each $i \in \{1, \ldots, a\}$ such that $c \in C_i$, there exists an edge cv of T with $v \in V_i$. Therefore, $\delta(c, T) \geq \delta(c,\text{BC}(G))$.

3 The graph as the union of its blocks

We will consider each problem in a separated subsection.

3.1 Minimum Leaves problem

The simplest case is that of ML. Let $\text{ML}^1(B_i)$ denote the problem of obtaining an ST T_i of B_i with as few leaves in $V_i \setminus C_i$ as possible. As stated in the forthcoming Proposition 1, the union of optimal solutions to $\text{ML}^1(B_i)$ for $i \in \{1, \ldots, a\}$ provides the optimal solution to ML(G).

Proposition 1 *If* $T_1, \ldots, T_a$ *are optimal solutions to* $\text{ML}^1(B_1)$,...,$\text{ML}^1(B_a)$ *respectively, then* $\bigcup_{i=1}^{a} T_i$ *is an optimal solution to* ML(G) *and* $v^*(\text{ML}(G))=\sum_{i=1}^{a} v^*(\text{ML}^1(B_i))$.

Proof From Remark 1, the articulation vertices have degree greater than one in any ST. Thus, the articulation vertices are not leaves in any spanning tree in G. Then, minimizing the number of leaves in $V \setminus C$ is equivalent to minimizing the number of leaves in V. □

3.2 Minimum Branch Vertices problem

Unfortunately, the resolution of the MBV problem cannot be done by simply bringing together the optimal trees corresponding to the blocks of the graph. A simple counterexample is shown in Fig. 2. The graph in Fig. 2a contains two blocks, the articulation vertex is the black one. The optimal value for each of the MBV subproblems on the blocks is one. Figure 2b, c show two pairs of optimal trees of the blocks. The union of optimal trees in Fig. 2b has one branch vertex, while this union in Fig. 2c has three branches, since in the gathered tree the articulation vertex has degree three.

The auxiliary problem $\text{MBV}^1(B_i, \gamma, c)$ has been designed to address with this drawback. For each $i \in \{1, \ldots, a\}$, $c \in C_i$ and a function γ defined as in Sect. 2, $\text{MBV}^1(B_i, \gamma, c)$ is defined as the optimization problem that intends to obtain an ST T_i of B_i which minimizes the number of branch vertices in $V_i \setminus C_i$ plus the number of

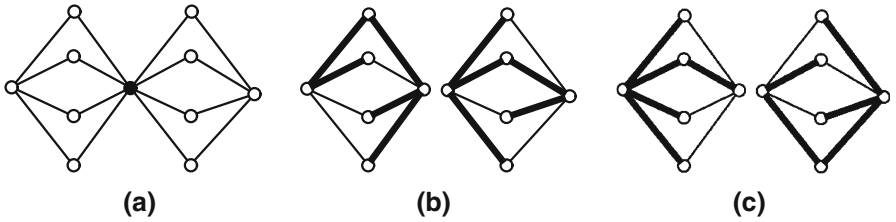

Fig. 2 Optimal subtrees providing one optimal and one non-optimal tree for the MBV

γ-branch vertices in C_i. In case there are optimal solutions for which c is a γ-branch vertex, $\text{MBV}^1(B_i, \gamma, c)$ asks for one of these.

Proposition 2 considers the case of a graph with only one articulation vertex, i.e., $C = \{c\}$. Here $a - 1$ refers the function γ with $\gamma(c) = a - 1$. The case $a \geq 3$ is straightforward. For proving the result when c splits G in two blocks, the number of branches of the union of one ST per block is computed. Since this number is minimal when the spanning trees on the blocks are the solutions to $\text{MBV}^1(B_i, a - 1, c)$, the result follows.

Proposition 2 *If $C = \{c\}$ and T_i is an optimal tree for $\text{MBV}^1(B_i, a - 1, c)$ for all $i \in \{1, \ldots, a\}$, then $\bigcup_{i=1}^{a} T_i$ is an optimal solution to $\text{MBV}(G)$.*

Proof From Remark 1, if $C = \{c\}$ and $a \geq 3$, the result trivially holds (c is a branch vertex of any spanning tree in G). Let then a be equal to 2, and let n_i and T_i, $i = 1, 2$, be the optimal value and the optimal tree of $\text{MBV}^1(B_i, 1, c)$. The number of branch vertices of $T_1 \cup T_2$ is the sum of both optimal values or this sum minus one, depending on the degree of c. In particular, it is

$$b = \begin{cases} n_1 + n_2 - 1 & \text{if } \delta(c, T_1) \geq 2 \text{ and } \delta(c, T_2) \geq 2, \\ n_1 + n_2 & \text{otherwise.} \end{cases}$$

Let T_i' be two other spanning trees of B_i with objective value in $\text{MBV}^1(B_i, 1, c)$ equal to $n_i' \geq n_i$, $i = 1, 2$. The number of branch vertices of $T_1' \cup T_2'$ is

$$b' = \begin{cases} n_1' + n_2' - 1 & \text{if } \delta(c, T_1') \geq 2 \text{ and } \delta(c, T_2') \geq 2, \\ n_1' + n_2' & \text{otherwise.} \end{cases}$$

If $n_1' > n_1$ or $n_2' > n_2$ then $b' \geq b$. If $n_i' = n_i$, $i = 1, 2$, then b' would be less than b iff $\delta(c, T_1') \geq 2$, $\delta(c, T_2') \geq 2$ and $\delta(c, T_1) \leq 1$ or $\delta(c, T_2) \leq 1$. However this is impossible because the optimization problem $\text{MBV}^1(B_i, 1, c)$ asks for solutions in which c is a 1-branch vertex. □

Note that $v^*(\text{MBV}(G))$ is not necessarily equal to $\sum_{i=1}^{a} v^*(\text{MBV}^1(B_i, a - 1, c))$. For instance, it could happen $\delta(c, T_1) = 3$ and $\delta(c, T_2) = 2$, in which case

$$v^*(\text{MBV}(G)) = v^*(\text{MBV}^1(B_1, a - 1, c)) + v^*(\text{MBV}^1(B_2, a - 1, c)) - 1.$$

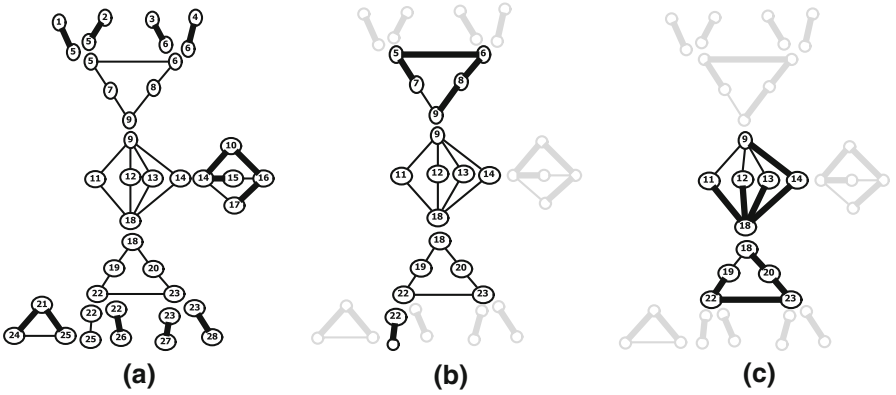

Fig. 3 Illustration of Algorithm 1. **a** First iteration, **b** second iteration, **c** third and last iteration

When G contains more than one articulation vertex, there will be blocks whose set of nodes includes more than one articulation vertex too. An iterative procedure is developed below that starts by solving auxiliary problems associated with the blocks which are leaves of the block-cutpoint graph of G and then sends the required information to the nearest blocks, so advancing towards the root of the tree. The optimal solution produced by the algorithm is the tree denoted by T^*.

Algorithm 1 1. Set $\hat{G} = G$, $U = \emptyset$ and $\gamma(v) = 1$ for all $v \in C$. Set $T^* = (V, \emptyset)$.
2. Let $\hat{B}_{i_1}, \ldots, \hat{B}_{i_t}$ be the leaf blocks of $\hat{G}$.
3. For $j = 1, \ldots, t$ do
   (a) If $t \geq 2$, let c be the articulation vertex adjacent to $\hat{B}_{i_j}$ in $\mathrm{BC}(\hat{G})$. Otherwise, let c be any cut vertex of $\hat{B}_{i_1}$. Do $\hat{G} = \hat{G} - (V(\hat{B}_{i_j}) \setminus \{c\})$.
   (b) Solve $\mathrm{MBV}^1(\hat{B}_{i_j}, \gamma, c)$. Let T_{i_j} be the optimal ST. Do $T^* = T^* \cup T_{i_j}$.
   (c) If $c \in U$, do $\gamma(c) = \gamma(c) + \delta(c, T_{i_j})$. Otherwise, do $\gamma(c) = \delta(c, T_{i_j})$.
   (d) Do $U = U \cup \{c\}$.
4. If $|V(\hat{G})| > 1$, go to Step 1.

Example 1 Let G be the graph in Fig. 1a. The algorithm takes three iterations for obtaining the optimal spanning tree. The three iterations are illustrated in Fig. 3a–c.

In the first iteration, $\hat{G}$ is still equal to G, and $\mathrm{BC}(G)$ contains nine leaf blocks (those represented in Fig. 1b), B1, B2, B3, B4, B7, B9, B11, B12 and B13, whose optimal trees in Step 2.c of Algorithm 1 are the graphs with bold edges in Fig. 3a. Consider, for instance, block B7. There are ten spanning trees of B7 that reach the optimal value of 1 in $\mathrm{MBV}^1(\mathrm{B7}, 1, 14)$. These ten trees are represented in Fig. 4. Note that node 14 contributes one unit to the objective if its degree is at least two, instead of three. However only in the seven trees of the first row of Fig. 4 the node 14 is a 1-branch vertex. Since all these seven trees are optimal and satisfy the required condition, any of them could have been produced by Algorithm 1. The first of them is the one in bold in Fig. 3a. After solving the subproblem, γ-values are updated to $\gamma(5) = \gamma(6) = \gamma(14) = \gamma(23) = 2$ and $\gamma(22) = \gamma(25) = 1$.

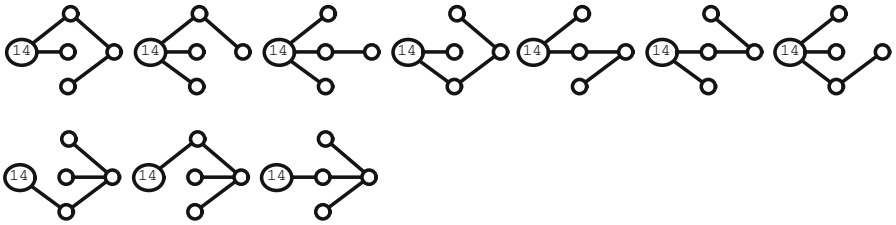

Fig. 4 All the optimal trees in B7

In the second iteration the gray blocks in Fig. 3b are removed from $\hat{G}$, that now contains only two leaf blocks, B5 and B10, whose optimal trees are the graphs with bold edges in Fig. 3b. Note that the effect of the γ-values is that the articulations contained in, for instance, B5 that will compulsorily be branches in the final solution are receiving more edges in the spanning tree of B5 than the articulations that can still be non-branch vertices. After solving this subproblem, γ-values are updated to $\gamma(9) = 1$, $\gamma(22) = 1 + 1$, indicating that node 22 will be necessarily an articulation in contrast to node 9 that, by the moment, only has degree one.

In the last iteration the gray blocks on Fig. 3c are removed from $\hat{G}$, that now contains again two leaf blocks, B6 and B8, whose optimal trees are the graphs with bold edges in Fig. 3c.

Finally, the optimal solution to MBV(G) is the union of the bold subtrees produced in the three iterations, and has optimal value 6.

Proposition 3 *The tree T^* obtained with Algorithm 1 is an optimal solution to* MBV(G).

Proof Let T' be an ST of G providing the best objective value for problem MBV and let T^* be the ST produced by Algorithm 1. Suppose that T' has less branch vertices than T^*.

Consider B, a leaf block of G containing articulation c, and replace in T' the edges with both extremes in $V(B)$ by the edges of T^* with both extremes in $V(B)$. The graph thus obtained is also a tree. Since T' is optimal, the number of branches of the new tree cannot be less than the number of branches of T'. If the number of branches is equal, remove $B - \{c\}$ from G and choose a new leaf block until the number of branches increases (choose the blocks in the order given in Algorithm 1). Taking into account that the edges of T^* in this block were obtained by solving the subproblem $\text{MBV}^1(B, \gamma, c)$, in such a way that the number of branches in B and the number of branches in the blocks B_i of BC(G) such that $B \prec B_i$ is minimized, it implies that c is not a 1-branch vertex. But this contradicts the fact that the optimal solution to $\text{MBV}^1(B, \gamma, c)$ is chosen so that c is a 1-branch vertex if possible. □

The complexity of Algorithm 1 is the complexity of finding the articulation vertices. Once the articulation vertices are identified the algorithm just solves one model at each block. For the identification of the articulation vertices the algorithm in [24] is used which is a linear-time algorithm based on single depth-first search.

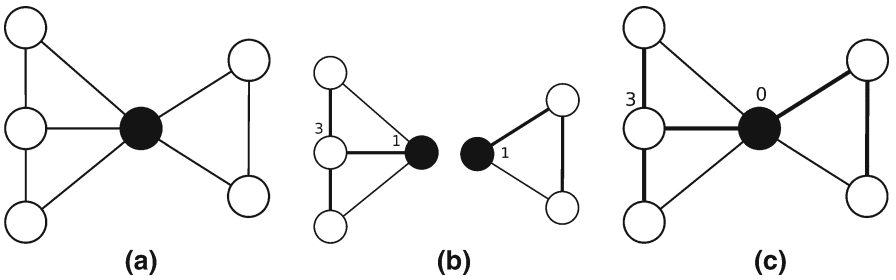

Fig. 5 Two subtrees of two blocks

3.3 Minimum Degree Sum problem

When the interest is on the degree sum of the branches, the analysis is a bit more difficult. Consider first a graph G with a unique articulation vertex c. If there are more than two blocks B_i, c is guaranteed to be a branch, and then its degree will be counted in the objective function. Consequently MDS(G) can be solved by blocks, simply using as objective function to be minimized at each block the degree of c plus the sum of the degrees of the branch vertices in $V_i \setminus \{c\}$. Then the union of the optimal STs of the blocks is an optimal solution to MDS(G) and the optimal value of MDS(G) is the sum of the optimal values of the subproblems. If, instead, removing c from G gives rise to exactly two connected components, this calculation will produce a wrong result. The degree sum of c plus the degree sum of the branches in $V \setminus \{c\}$ might not be the degree sum of the union spanning tree, as shown in Fig. 5. In (a), a graph is shown where the black node is the articulation vertex. In (b), an ST at each block is shown with thick edges. In the left hand block of (b), there is one branch with degree three and the degree of c is one, totaling 4. In the right hand block of (b) there are no branches and the degree of c is one, totaling 1. The addition of the degree sum of branches plus the degree sum of c gives 5. But in the joint tree represented in (c), c is not a branch and then the degree sum of branches is 3. Then we have to take into account that, only when the degree of c in the STs of both blocks is 1, we must subtract two units to the sum produced by the above procedure. Consequently, we must distinguish two cases when solving separately the blocks. Since the blocks are going to be sequentially solved, from the leaf blocks to the root of the block-cutpoint graph, when solving each block two subproblems need to be solved, one considering that the articulation that links the leaf block with the rest of the graph will finally have degree one in the solution of the neighbor block, and another one considering that it will have degree at least two.

To manage this disjunction in the design of the algorithm, the following two auxiliary problems are defined for each $i \in \{1, \ldots, a\}$, $c \in C_i$ and $F, D \subseteq C_i \setminus \{c\}$ such that $F \cap D = \emptyset$:

- MDS$^1(B_i, F, D, c)$ is an optimization problem aiming to obtain an ST T_i of B_i which minimizes the sum of
  (i) the degrees of the branches in $V_i \setminus C_i$,
  (ii) the degrees of the vertices in C_i,

(iii) minus twice the number of leaves in F,
(iv) minus the number of leaves in D. In the event that there are optimal solutions for which $\delta(c, T_i) = 1$, $\text{MDS}^1(B_i, F, D, c)$ asks for one of these.

- $\text{MDS}^2(B_i, F, D, c)$ is the problem of obtaining an ST T_i of B_i that minimizes (i)+ (ii)+ (iii)+ (iv) as above, but constraining the degree of c to take value 1.

None of the aforementioned problems can be infeasible. In particular, since $B_i = (V_i, E_i)$ is 2-connected, there exists an ST $T_i = (V_i \setminus \{c\}, E_{T_i})$ of $B_i - \{c\}$ and then the tree $(V_i, E_{T_i} \cup \{cv\})$ is a feasible solution to $\text{MDS}^2(B_i, F, D, c)$ for any $cv \in E_i$.

Algorithm 2 states the procedure for the general case with any number of articulation vertices and any number of blocks. Generally speaking, in Algorithm 2 the mentioned auxiliary problems are solved recursively from the leaf blocks of $\hat{G}$ towards the root block. $\text{MDS}^1(B_i, F, D, c)$ is solved at each block while $\text{MDS}^2(B_i, F, D, c)$ is solved only at some of them. The function $\theta : C \to \{0, 1\}$ indicates if the problem $\text{MDS}^2(B_i, F, D, c)$ was solved for the block(s) adjacent to c in $\text{BC}(\hat{G})$ ($\theta(v) = 1$ indicates that it was).

Algorithm 2 1. Set $\hat{G} = G$, $\hat{T}_1, \ldots, \hat{T}_a = (V, \emptyset)$, and $\gamma(v) = \theta(v) = 0$ for all $v \in C$.
2. Let $\hat{B}_{i_1}, \ldots, \hat{B}_{i_t}$ be the leaf blocks of $\hat{G}$.
3. For $j = 1, \ldots, t$ do:
   (a) If $t \geq 2$, let c be the articulation vertex adjacent to $\hat{B}_{i_j}$ in $\text{BC}(\hat{G})$. Otherwise, let c be any cut vertex of $\hat{B}_{i_1}$.
   (b) Do $\hat{G} = \hat{G} - V(\hat{B}_{i_j} \setminus \{c\})$.
   (c) Do $F = \{v \in C_{i_j} : \gamma(v) = 1\}$ and $D = \{v \in C_{i_j} : \gamma(v) = 2, \theta(v) = 1\}$. Solve $\text{MDS}^1(\hat{B}_{i_j}, F, D, c)$. Let $T^1_{i_j}$ be an optimal ST and let v^*_1 be the optimal value.
   (d) Do $\gamma(c) = \gamma(c) + \delta(c, T^1_{i_j})$.
   (e) If $\delta(c, \text{BC}(G)) = 2$ and $\delta(c, T^1_{i_j}) \geq 2$:
      (i) Do $F = \{v \in C_{i_j} : \gamma(v) = 1\}$, $D = \{v \in C_{i_j} : \gamma(v) = 2, \theta(v) = 1\}$. Solve $\text{MDS}^2(B_{i_j}, F, D, c)$. Let $T^2_{i_j}$ be the optimal spanning tree and v^*_2 be the optimal value.
      (ii) If $v^*_2 = v^*_1 + 1$, do $\theta(c) = 1$.
      Otherwise do $\hat{T}_{i_j} = T^1_{i_j}$.
4. If $|V(\hat{G})| > 1$, go to Step 2.
5. For each pair of blocks B_f, B_h such that $\hat{T}_f = \emptyset$, $\hat{T}_h \neq \emptyset$, $B_f \cap B_h = \{v\}$, and $h < f$ in $\text{BC}(G)$ do

$$\hat{T}_f = \begin{cases} T^1_f & \text{if } \delta(v, \hat{T}_h) \geq 2 \\ T^2_f & \text{otherwise.} \end{cases}$$

6. The union spanning tree $T = \bigcup_{i=1}^{k} \hat{T}_i$ is an optimal solution to $\text{MDS}(G)$.

How Algorithm 2 works is now shown with an example.

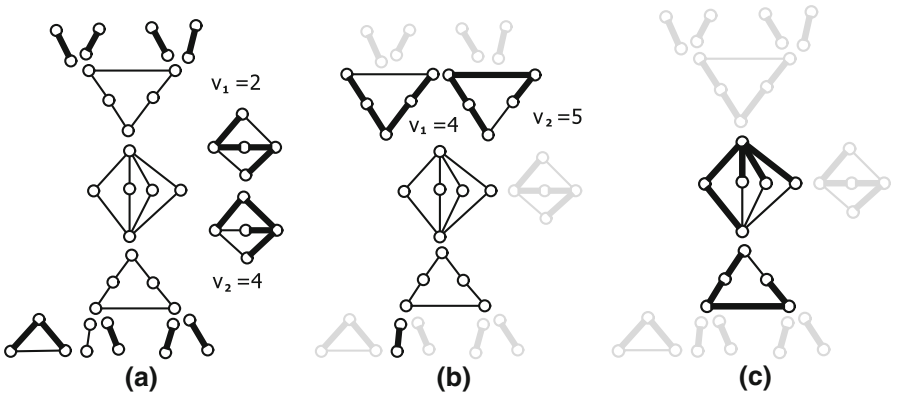

Fig. 6 Illustration of Algorithm 2. **a** First iteration, **b** second iteration, **c** third and last iteration

Example 2 Consider the graph of Fig. 1a. Algorithm 2 takes three iterations for obtaining the optimal spanning tree. In the first iteration, the first block to be considered is B1. Since the block only contains one edge, the optimal solution to MDS^1(B1,∅, ∅, {5}) is trivial. Then $\gamma(5) = 1$. Since the degree of node 5 in the block-cutpoint tree is 3, Step (e) is not applied. Considering then B2, $F = \{5\}$ since the previous degree of node 5 was 1. Now, MDS^1(B2,{5}, ∅, {5}) has again a trivial solution, $\gamma(5) = 2$ and Step (e) is not applied. Blocks 3 and 4 are similarly treated and $\gamma(6) = 2$. Let now focus the attention on block B7. Figure 6a shows two trees for the block B7. The first one is optimal to MDS^1(B7,∅, ∅, {14}) and $\gamma(14) = 2$. To see it, consider again Fig. 4 showing all the spanning trees in B7, whose objective values in MDS^1(B7,∅, ∅, {14}) are (from top to bottom and from left to right) 2, 3, 3, 2, 2, 2, 3, 4, 4, 4. All those with objective value equal to 2 are optimal solutions. Since articulation 14 corresponds only to 2 blocks, now condition (e) is satisfied and MDS^2(B7,∅, ∅, {14}) has to be solved. All the trees in the second row of Fig. 4 give the optimal value of 4, and one of them has been drawn in Fig. 6a, under the previous one. Since $v_1^* = 2$ and $v_2^* = 4$, then $\theta(14) = 0$. Here the usefulness of this parameter appears: A solution to the second subproblem could be better than a solution to the first one if finally articulation 14 would have degree 1 in block B6, because in such a solution 14 would not be a branch, and its contribution to the objective would be null. But in this case, the saving of one unit would be at the expense of an increment of two units, and then this possibility is discarded by fixing $\theta(14) = 0$.

On the contrary, going straight to the second iteration, condition in Step (e) of Algorithm 2 holds for B5, since $v_2^* = v_1^* + 1 = 5$ and then $\theta(9) = 1$, meaning that both sub-solutions should be considered. When solving subsequent subproblems, F and D will be considered to discount the extra cost of the optimal solutions of the previous blocks that finally will only receive one edge from the current subproblem, avoiding the articulation to be a branch.

In the last iteration, the solution of the last blocks determines which of the previous subtrees belong to the final solution. Observe that v^*(MDS^1(B6, ∅, {9, 18}, {14})) = $4 + 2 - 1 - 1 = 3$. Figure 7 shows all the relevant trees of B6. Their objective value

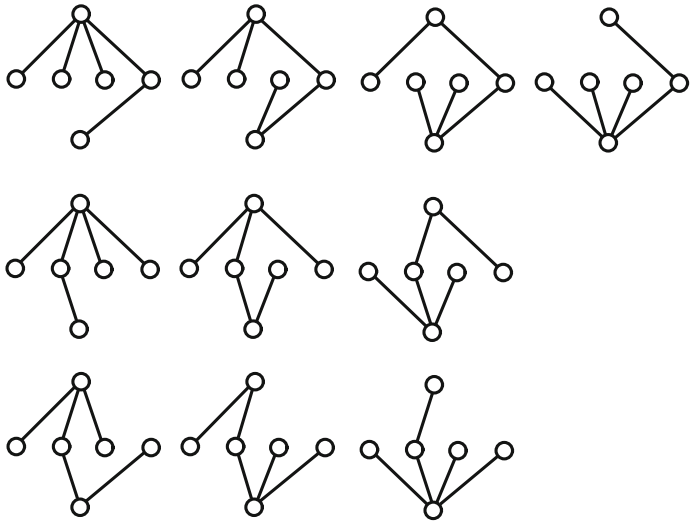

Fig. 7 All the relevant trees of B6

in $\text{MDS}^1(\text{B6}, \emptyset, \{9, 18\}, \{14\})$ is obtained as the degree of node 9 plus the degree of node 14 plus the degree of node 18 and minus one if the degree of nodes 9 or 18 is one. Concretely, (from top to bottom and from left to right) 6, 7, 7, 6, 5, 6, 6, 6, 6, 5. The lowest value is 5 and both trees whose objective value is 5 are optimal.

Proposition 4 *The tree T^* obtained with Algorithm 2 is an optimal solution to* $\text{MDS}(G)$.

Let T' be an ST of G providing the best objective value for problem MDS and let T^* be the ST produced by Algorithm 2. Assume that T' has smaller degree sum than T^*.

Consider B, a leaf block of G containing articulation c, and replace in T' the edges with both extremes in $V(B)$ by the edges of T^* with both extremes in $V(B)$. The graph thus obtained is also a tree. Since T' is optimal for MDS, the degree sum of branches of the new tree cannot be less than the degree sum of branches of T'. If the degree sum is equal, remove $B - \{c\}$ from G and choose a new leaf block until the degree sum increases (choose the blocks in the order given in Algorithm 2). Taking into account that the edges of T^* in this block were obtained by following Algorithm 2, in such a way that the degree sum in B and the degree sum in the blocks B_i of $\text{BC}(G)$ such that $B \prec B_i$ is minimized, it implies that the number of edges $cv \in E(B) \cap T'$ was one and the number of edges $cv \in E(B) \cap T^*$ is more than one. But this contradicts the fact that the optimal solution to $\text{MDS}^1(B, F, D, c)$ is chosen so that c is a leaf in case of tied optimal solutions. □

The complexity of Algorithm 2 is also the complexity of finding the articulation vertices and for the identification of the articulation vertices the algorithm in [24] is used which is a linear-time algorithm based on single depth-first search.

4 Formulations

Formulations of the problems ML, MBV and MDS are required in order to implement the decomposition algorithms described in the previous section. Since the goal of this paper is to solve optimization problems for the blocks of the graph and then to compute the optimal solution of the whole graph as the union of certain spanning trees of the blocks, it is necessary to formulate $\mathrm{ML}^1(B_i)$, $\mathrm{MBV}^1(B_i, \gamma, c)$, $\mathrm{MDS}^1(B_i, F, D, c)$ and $\mathrm{MDS}^2(B_i, F, D, c)$. Nevertheless, these problems can be formulated modifying properly the formulations of ML, MBV and MDS. The so-called Single Commodity Formulations in [3] will be used as a starting point.

The formulations are defined on directed graphs, thus a directed version of G that contains both arcs for each edge is considered. Each edge $uv \in E$ is split into two arcs (u, v) and (v, u) and this set of arcs is called A. Analogously, A_i is the set of arcs obtained splitting the edges in E_i for all $i \in \{1, \ldots, a\}$. A node $s \in V$ is arbitrarily chosen to be the root of the arborescence produced by the formulations.

All the formulations in this section share the families of x-variables and f-variables. The standard binary decision variable x_{uv} takes value one when (u, v) is an arc of the arborescence, $(u, v) \in A$. Variable f_{uv} represents the flow traversing (u, v), for all $(u, v) \in A$, in such a way that there will be a flow of $|V| - 1$ units with origin in the root node s.

4.1 Minimum Leaves problem

In the Single Commodity Formulations of $\mathrm{ML}(G)$, the binary variable w_v takes value one when v is a leaf, $v \in V$.

$$\mathrm{FML}(G) \qquad \min \sum_{v \in V} w_v$$

$$\text{s.t.} \sum_{\substack{u \in V: \\ (u,v) \in A}} x_{uv} = 1 \quad \forall v \in V \setminus \{s\} \tag{1}$$

$$\sum_{\substack{v \in V: \\ (s,v) \in A}} f_{sv} - \sum_{\substack{v \in V: \\ (v,s) \in A}} f_{vs} = |V| - 1 \tag{2}$$

$$\sum_{\substack{u \in V: \\ (u,v) \in A}} f_{uv} - \sum_{\substack{u \in V: \\ (v,u) \in A}} f_{vu} = 1 \quad \forall v \in V \setminus \{s\} \tag{3}$$

$$x_{uv} \leq f_{uv} \quad \forall (u, v) \in A \tag{4}$$

$$\sum_{\substack{u \in V: \\ (v,u) \in A}} x_{vu} + \sum_{\substack{u \in V: \\ (u,v) \in A}} x_{uv} + w_v \geq 2 \quad \forall v \in V \tag{5}$$

$$x_{uv} + x_{vu} \leq 1 \quad \forall uv \in E \tag{6}$$

$$w_v \in \{0, 1\} \quad \forall v \in V \tag{7}$$

$$x_{uv} \in \{0, 1\} \quad \forall (u, v) \in A \tag{8}$$

$$f_{uv} \geq 0 \quad \forall (u, v) \in A. \tag{9}$$

The objective function of FML(G) is the number of leaves in the ST. Constraints (1) state that for each $v \in V$, other than the root node, there is exactly one arc which points v. Constraints (2) and (3) keep the connectivity. Constraints (4) relate x-variables with f-variables: If there is no flow between two nodes, the associated arc does not belong to the arborescence. Constraints (5) force variable w_v to be one if the degree of node v is one, i.e., it is a leaf. Finally, constraints (6) are valid inequalities that are always satisfied by the solution, since the two arcs associated to the same edge cannot be in the arborescence simultaneously.

The related auxiliary problem $\mathrm{ML}^1(B_i)$, which is the problem of obtaining an ST T_i of B_i with as few leaves in $V_i \setminus C_i$ as possible, can be formulated as follows, taking into account that constraints (1)–(4) and (6)–(9) have to be slightly modified since the graph of application is B_i instead of G, i.e., that V, A and E have to be replaced by V_i, A_i and E_i respectively in (1)–(4) and (6)–(9).

$$\begin{aligned} \mathrm{FML}^1(B_i) \qquad & \min \sum_{v \in V_i \setminus C_i} w_v \\ & \text{s.t. (1)–(4), (6)–(9)} \\ & \sum_{\substack{u \in V_i: \\ (v,u) \in A_i}} x_{vu} + \sum_{\substack{u \in V_i: \\ (u,v) \in A_i}} x_{uv} + w_v \geq 2 \quad \forall v \in V_i \setminus C_i. \end{aligned} \tag{10}$$

Constraints (5) have been replaced by constraints (10) which assign value one to variable w_v only if v is not an articulation vertex and it is a leaf.

4.2 Minimum Branch Vertices problem

In the following, the binary variable y_v takes value one when v is a branch vertex, $v \in V$.

$$\begin{aligned} \mathrm{FMBV}(G) \qquad & \min \sum_{v \in V} y_v \\ & \text{s.t. (1)–(4), (6), (8), (9)} \\ & \sum_{\substack{u \in V: \\ (v,u) \in A}} x_{vu} + \sum_{\substack{u \in V: \\ (u,v) \in A}} x_{uv} \leq \delta(v, G) y_v + 2 \quad \forall v \in V \end{aligned} \tag{11}$$

$$y_v \in \{0, 1\} \quad \forall v \in V. \tag{12}$$

The objective function of FMBV(G) is the number of branch vertices. Each constraint (11) forces a variable y_v to be one if the degree of node v in the tree is greater than two.

The related auxiliary problem $\mathrm{MBV}^1(B_i, \gamma, c)$, which is the problem of obtaining an ST T_i of B_i minimizing the number of branch vertices in $V_i \setminus C_i$ plus the number of γ-branch vertices in C_i and that, in case there are optimal solutions for which c

is a γ-branch vertex, asks for one of these, can be formulated as follows. Note that constraints (1)–(4), (6), (8), (9) and (12) have to be adapted to the subgraph where the problem is being solved, i.e., block B_i.

$$\text{FMBV}^1(B_i, \gamma, c) \qquad \min\ \alpha y_c + \sum_{v \in V_i \setminus \{c\}} y_v$$
$$\text{s.t. (1)–(4), (6), (8), (9), (12)}$$
$$\sum_{\substack{u \in V_i: \\ (v,u) \in A_i}} x_{vu} + \sum_{\substack{u \in V_i: \\ (u,v) \in A_i}} x_{uv} \leq \delta(v, B_i) y_v + 2 \quad \forall v \in V_i \setminus C_i \tag{13}$$
$$\sum_{\substack{u \in V_i: \\ (v,u) \in A_i}} x_{vu} + \sum_{\substack{u \in V_i: \\ (u,v) \in A_i}} x_{uv} \leq \delta(v, B_i) y_v + 2 - \gamma(v) \quad \forall v \in C_i \tag{14}$$

where $\alpha \in (0, 1)$.

Due to constraints (13), variable y_v takes value one when the degree of node v is greater than 2 (branch). The degree of an articulation v needs to be greater than $2-\gamma(v)$ (γ-branch) and constraints (14) are devoted to this. The coefficient α of y_c breaks the ties in favor of the solutions (x, f, y) to $\text{FMBV}^1(B_i, \gamma, c)$ with $y_c = 1$.

According to Algorithm 1, formulation $\text{FMBV}^1(B_i, \gamma, c)$ is used for leaf blocks of the block-cutpoint graph BC(G).

4.3 Minimum Degree Sum problem

In this subsection, variable z_v is the degree of a branch vertex v, $v \in V$.

$$\text{FMDS}(G) \qquad \min \sum_{v \in V} z_v$$
$$\text{s.t. (1)–(4), (6), (8), (9), (11), (12)}$$
$$\sum_{\substack{u \in V: \\ (v,u) \in A}} x_{vu} + \sum_{\substack{u \in V: \\ (u,v) \in A}} x_{uv} \leq z_v + 2 - 2y_v \quad \forall v \in V \tag{15}$$
$$z_v \geq 0 \quad \forall v \in V. \tag{16}$$

The objective function of FMDS(G) is the sum of the degrees of all the branch vertices. Each constraint (15) makes z_v equal to the degree of node v if it is larger than two ($y_v = 1$).

The auxiliary problem $\text{MDS}^1(B_i, F, D, c)$, which is the problem of obtaining an ST T_i of B_i minimizing the sum of (i) the degrees of the branch vertices in $V_i \setminus C_i$, (ii) the degrees of the vertices in C_i, (iii) minus twice the number of leaves in F, (iv) minus the

number of leaves in D and, in case there are optimal solutions for which c is a leaf, asks for one of these, can be formulated as follows.

$$\text{FMDS}^1(B_i, F, D, c)$$

$$\min \; -\alpha(1-y_c) + \sum_{v\in V_i} z_v - 2\sum_{v\in F}(1-y_v) - \sum_{v\in D}(1-y_v)$$

$$\text{s.t. } (1)\text{–}(4), (6), (8), (9), (11), (12), (13), (16)$$

$$\sum_{\substack{u\in V_i:\\ (v,u)\in A_i}} x_{vu} + \sum_{\substack{u\in V_i:\\ (u,v)\in A_i}} x_{uv} \le \delta(v, B_i)y_v + 1 \qquad \forall v \in C_i \tag{17}$$

$$\sum_{\substack{u\in V_i:\\ (v,u)\in A_i}} x_{vu} + \sum_{\substack{u\in V_i:\\ (u,v)\in A_i}} x_{uv} \le z_v + 2 - 2y_v \qquad \forall v \in V_i \setminus C_i \tag{18}$$

$$\sum_{\substack{u\in V_i:\\ (v,u)\in A_i}} x_{vu} + \sum_{\substack{u\in V_i:\\ (u,v)\in A_i}} x_{uv} \le z_v \qquad \forall v \in C_i \tag{19}$$

where $\alpha \in (0, 1)$. Each constraint (17) forces variable y_v to take value one when the degree of the node v in the solution ST is greater than 1. Consequently, $(1 - y_v)$ takes the value 1 when v is a leaf of the ST, since it has negative coefficient in the objective function. Here (18) is a subset of constraints (15) for $v \in V_i \setminus C_i$, i.e., makes z_v equal to the degree of node v if it is greater than two for all $v \in V_i \setminus C_i$. Each constraint (19) makes z_v equal to the degree of node v for all $v \in C_i$. Thus the addend $\sum_{v\in V_i} z_v$ in the objective function is the sum of the degrees of the branch vertices in $V_i \setminus C_i$ and the degrees of all vertices in C_i. On the other hand, the addend $-2\sum_{v\in F}(1 - y_v) - \sum_{v\in D}(1 - y_v)$ is the sum of minus twice the number of leaves in F and minus the number of leaves in D. The addend $-\alpha(1 - y_c)$ makes that among the solutions (x, f, y, z) to $\text{FMDS}^1(B_i, F, D, c)$, those with $y_c = 0$ are preferred.

The auxiliary problem $\text{MDS}^2(B_i, F, D, c)$ is obtained from $\text{FMDS}^1(B_i, F, D, c)$ constraining the degree of c to be 1 and removing from the objective function the first addend, since it becomes constant. Here, constraints (1)–(4), (6), (8)–(9), (12), (13), (16) and (17)–(19) concern only B_i.

$$\text{FMDS}^2(B_i, F, D, c) \qquad \min \sum_{v\in V_i} z_v - 2\sum_{v\in F}(1-y_v) - \sum_{v\in D}(1-y_v)$$

$$\text{s.t. } (1)\text{–}(4), (6), (8)\text{–}(9), (11), (12), (13), (16), (17)\text{–}(19)$$

$$\sum_{(c,u)\in A_i} x_{cu} + \sum_{(u,c)\in A_i} x_{uc} = 1.$$

4.4 Lower bounds from the linear relaxations

Combining the linear relaxations of the auxiliary problems, lower bounds for the three problems ML, MBV and MDS can be obtained. Let $\text{FML}(B_i)_{LR}$ be the linear relaxation of $\text{FML}^1(B_i)$. The lower bound for the optimal value of $\text{ML}(G)$ is calculated as

$$\sum_{i=1}^{a} \lceil v^*(\text{FML}(B_i)_{LR}) \rceil. \tag{20}$$

The rounding up of $v^*(\text{FML}(B_i)_{LR})$ is calculated because the number of leaves is an integer value. The addition of all the roundings is a lower bound because the articulation vertices are never leaves of an ST.

Applying Remark 1, all the articulation vertices whose degree in $\text{BC}(G)$ is greater than 2 are branch vertices of any ST. Also the articulation vertices which are extremes of bridges are 1-branch vertices of any ST. Then, the lower bound on the optimal value of $\text{FMBV}(G)$ is the number of articulation vertices whose degree in $\text{BC}(G)$ is larger than 2 plus the sum of roundings of the associated linear problems.

Let $\text{FMBV}(B_i, \gamma, c)_{LR}$ be the linear relaxation of $\text{FMBV}^1(B_i, \gamma, c)$. In order to solve it, three modifications are introduced in Algorithm 1: (i) $\gamma(v)$ is initialized to $-|V_i|$ for all $v \in C$ such that $\delta(v,\text{BC}(G)) \geq 3$ and v^* is initialized to zero, (ii) Step 3(b) is replaced by "Solve $\text{FMBV}(B_{i_j}, \gamma, c)_{LR}$. Do $v^* = v^* + \lceil v^*(\text{FMBV}(B_i, \gamma, c)_{LR}) \rceil''$ and (iii) Step 3(c) is replaced by "If c is not the extreme of a bridge, do $\gamma(c) = -|V_i|$". Let v^* be the output of the modified algorithm. The lower bound is

$$|\{v \in C : \delta(v, \text{BC}(G)) \geq 3\}| + v^*. \tag{21}$$

Let $\text{FMDS}(B_i, F, D, c)_{LR}$ be the linear relaxation of $\text{FMDS}^1(B_i, F, D, c)$. Doing $F = \emptyset$ and $D = \{v \in C : \delta(v,\text{BC}(G)) = 2\}$, c any vertex in V_i and $\alpha = 0$, the lower bound on the optimal value of $\text{FMDS}(G)$ is

$$\sum_{i=1}^{a} \lceil v^*(\text{FMDS}(B_i, F, D, c)_{LR}) \rceil. \tag{22}$$

For this definition of the sets F and D, the problem $\text{MDS}^1(B_i, F, D, c)$ consists in obtaining an ST T_i of B_i which minimizes the sum of (i) the degrees of the branch vertices in $V_i \setminus C_i$, (ii) the degrees of the vertices v in C_i such that $\delta(v,\text{BC}(G)) \geq 3$ and (iii) the degrees of the 1-branch vertices v in C_i such that $\delta(v,\text{BC}(G)) = 2$.

5 Computational experiments

The experimental results in this section illustrate the benefit of decomposing $\text{ML}(G)$, $\text{MBV}(G)$ and $\text{MDS}(G)$ in subproblems based on the blocks of G.

There are several benchmark instances frequently used for checking the performance of different algorithms designed to solve ML, MDS and MBV. In order to check the performance of our algorithms with graphs having several large blocks and being easily reproducible, the instances in [2] have been combined as explained below. The number of

nodes n of the benchmark instances introduced and solved in [2] range between 20 and 500. For each value of n there are five graphs with different densities (number of edges m). And for each combination of n and m there are five different instances. We call *family* each of these sets of five instances of the same size.

Since the articulation vertices in the instances taken from [2] are mostly extremes of bridges whose other extreme has degree one, new instances have been generated from them in order to check the performance of the algorithms when there are articulation vertices with degree two in the block-cutpoint graph and which split the graph in blocks with similar number of nodes. This is the kind of graph where the decomposition method we have designed can take advantage.

We have linked the first three instances of each family to produce a larger graph with $3n$ nodes and $3m$ edges, and the last two graphs to produce another graph with $2n$ nodes and $2m$ edges. Linking two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $|V_1| = |V_2| = n$ to produce the linked graph $L(G_1, G_2)$ is done by identifying vertex $k = \lfloor n/2 \rfloor$ of G_1 with vertex $\ell = 1$ of G_2, unless some of them is an articulation vertex, in which case it is replaced by the next node in the list. The precise steps follow.

- Set $k = \lfloor n/2 \rfloor$ and $\ell = 1$.
- If i_k is an articulation vertex of G_1, do $k = k + 1$ until it is not.
- If j_ℓ is an articulation vertex of G_2, do $\ell = \ell + 1$ until it is not.
- Rename the node j_ℓ in G_2 as i_k.
- Do $L(G_1, G_2) = G_1 \cup G_2$.

Linking three graphs G_1, G_2 and G_3 has been done by calculating $L(L(G_1, G_2), G_3)$. The characteristics of the linked graphs generated with this procedure are summarized in Table1. Each line represents five different instances, thus in total the testbed consists of 140 instances. The number of nodes of the graph is given in column $|V|$, the number of linked graphs (2 or 3) is shown in column *Linked*, columns *Blocks*, *Bridges* and *Leaves* are the averages of these structures in the graphs. Note that here "leaf" refers to a node of degree one in the graph. Although all the leaves are extremes of bridges, not all the bridges incide in a leaf. Most of the blocks contain only two nodes, i.e., they are defined by bridges. Column *Easy Blocks* gives the averages of blocks which do not have articulation vertices with degree two in the block-cutpoint graph and thus do not require re-optimization in Algorithm 2.

The algorithms have been coded in C++ on a PC with a 2.33 GHz Intel Xeon dual core processor, 8.5 GB of RAM, and operating system linux debian 4.0. The optimization engine CPLEX v11.0 was used and default values were set for all but one parameters. The modified parameter is the one that allows to discard a node of the branching tree when the difference between its lower bound and the incumbent is less than or equal to a given value. This was fixed to 0.99 since the optimal values of the three problems studied in this paper are always integer.

Articulation vertices were identified following the algorithm given in [24]. On the other hand, Algorithms 1 and 2 were not applied to blocks containing only two nodes (i.e., bridges), since they have only one feasible solution. In this case, the edge was incorporated to the optimal tree without calling the solver.

Table 2 shows the reduction of time when solving ML(G) by decomposing G into blocks. v^* is the average optimal value and *Lower Bound* is the lower bound obtained using (20). The times (in seconds) for solving the instances with the formulation FML(G) and for solving ML $^1(B_i)$ for all $i \in \{1, \ldots, a\}$ (Proposition 1) are given in the last two

Table 1 Instances description

$\|V\|$	Linked	Blocks	Easy blocks	Bridges	Leaves
40	2	6.2	3.8	3.8	4.2
60	3	8.4	6.4	5.2	5.0
80	2	14.0	10.8	10.8	12.0
120	3	23.8	19.8	18.6	20.0
200	2	62.2	53.2	55.4	57.2
240	2	81.4	70.4	76.6	75.0
280	2	99.2	83.0	94	90.4
300	3	98.2	83.0	91	89.2
320	2	114.8	98.8	108	106.2
360	3	124.0	107.6	116.8	112.4
360	2	138.2	117.2	129.8	127.2
400	2	154.2	127.2	149	142.4
420	3	145.0	124.8	137.8	133.2
480	3	174.8	144.2	168.8	161.6
500	2	211.0	178.8	202.2	192.8
540	3	203.2	170.0	190.8	183.8
600	3	236.8	200.6	226.4	216.4
600	2	271.6	225.2	263.2	243.4
700	2	321.2	265.6	309.2	288.0
750	3	317.0	267	304.2	288.2
800	2	395.2	325.6	387.6	350.0
900	3	411.4	342.4	401	370.8
900	2	443.6	372.0	433.2	394.0
1000	2	510.4	422.8	499.2	449.0
1050	3	492.2	410.4	479.4	437.4
1200	3	588.6	480.6	573.8	520.0
1350	3	666.6	551.0	650.6	591.2
1500	3	774.4	636.2	758.4	680.8

columns. Decomposition times are always less than formulation times and, in average, the mean time is reduced from 8.5 to 2.6 s. On the other hand, Table 2 shows that the bound obtained in (20) is tight, and better than the column *Leaves* in Table1 which is also a naive lower bound on the optimal value of ML(G).

Table 3 shows the reduction of time when solving MBV(G) by decomposing G into blocks (following Algorithm 1). Here *Lower Bound* is the lower bound obtained using (21). First, the average time in seconds for solving MBV(G) by FMBV(G) is presented. Superindices represent the number of unsolved instances using FMBV(G) among the five instances considered in each row, within the time limit of one hour. There are only 109 instances among the 140 that could be solved with FMBV(G). The time in seconds for solving the same 109 *easy* instances applying Algorithm 1 is then shown, with a reduction of one order of magnitude. Moreover, applying Algorithm 1 all the 140 instances could be solved within a time limit of one hour, requiring a time in seconds shown in the last column.

Table 2 Results for the Minimum Leaves problem

$\|V\|$	Linked	υ^*	Lower bound	Time (s)	
				FML(G)	Blocks
40	2	6.2	5.8	0.2	0.4
60	3	8.4	8.4	0.0	0.2
80	2	15.2	15.0	0.2	0.2
120	3	22.6	22.2	1.0	0.4
200	2	60.8	60.8	1.2	0.8
240	2	77.8	77.8	1.4	0.6
280	2	93.6	93.6	1.6	0.8
300	3	93.8	93.6	3.0	0.8
320	2	110.4	110.4	2.0	1.4
360	3	119.4	119.0	2.8	1.2
360	2	131.2	131.2	2.0	1.4
400	2	144.4	144.4	2.2	1.0
420	3	140.0	140.0	2.6	0.4
480	3	165.2	165.2	5.4	1.2
500	2	195.2	195.2	4.2	1.6
540	3	191.8	191.8	5.2	2.4
600	3	219.6	219.4	9.0	2.2
600	2	244.4	244.4	1.6	2.4
700	2	289.0	288.8	9.4	4.2
750	3	293.2	293.0	17.8	2.8
800	2	351.0	351.0	10.0	3.2
900	3	372.4	372.4	32.0	5.8
900	2	395.4	395.2	14.0	6.4
1000	2	449.8	449.6	12.6	6.4
1050	3	441.0	441.0	22.6	5.0
1200	3	521.6	521.4	15.4	4.4
1350	3	592.8	592.8	8.5	7.4
1500	3	682.4	682.4	50.8	7.2
Averages		229.6	229.5	8.5	2.6

The 31 instances unaffordable for FMBV(G) were solved with Algorithm 1 keeping the average time below 1 min.

Table 4 shows the reduction of time when solving MDS(G) by decomposing G into blocks (following Algorithm 2). Here *Lower Bound* is the lower bound obtained using (22). When the graph is not decomposed into blocks, 14 of the 140 instances could not be solved within the time limit of one hour by using FMDS(G). The average time for solving the remaining 126 instances with FMDS(G) was 248.1 s while the average time for solving the same 126 instances following Algorithm 2 was 15.7 s. The average time of solving the 140 instances with Algorithm 2 was 22.6 s. Assuming that the time for solving the 14

Table 3 Results for the Minimum Branch Vertices problem

$\|V\|$	Linked	v^*	Lower bound	Time (s)		
				FMBV(G)	Algorithm 1	
					Easy	All
40	2	1.8	1.6	0.2	0.0	0.0
60	3	2.2	1.6	0.0	0.0	0.0
80	2	5.8	3.6	1.0	0.0	0.0
120	3	8.0	6.4	0.4	0.4	0.4
200	2	25.0	18.0	7.2	2.8	2.8
240	2	34.6	27.0	15.4	3.8	3.8
280	2	41.6	31.2	73.4	4.6	4.6
300	3	39.4	29.2	33.2	2.8	2.8
320	2	49.8	37.6	704.8	6.6	6.6
360	3	51.0	37.6	450.4	5.2	5.2
360	2	58.4	45.0	37.0	8.0	8.0
400	2	64.6	50.6	274.0	9.4	9.4
420	3	61.6	44.4	162.6	6.8	6.8
480	3	73.6	56.0	226.4	7.6	7.6
500	2	89.0	71.8	183.2	20.0	20.0
540	3	84.8	61.4	^{1}280.5	28.0	30.4
600	3	96.4	77.0	^{1}146.0	8.0	12.6
600	2	114.2	92.0	^{1}527.0	14.0	44.2
700	2	135.6	107.8	^{2}568.0	52.0	68.8
750	3	132.0	106.2	^{2}317.0	14.3	27.4
800	2	163.8	136.8	^{2}551.6	25.6	74.4
900	3	170.2	141.2	^{2}110.0	19.0	41.6
900	2	187.6	153.4	^{3}34.5	15.0	147.8
1000	2	211.0	179.4	^{2}216.6	36.0	134.8
1050	3	205.8	164.6	^{4}439.0	7.0	145.2
1200	3	244.8	201.8	^{4}71.0	5.0	63.6
1350	3	279.6	232.8	^{3}2887.0	14.5	213.0
1500	3	321.6	270.0	^{4}98.0	7.0	117.4
Averages		105.5	85.2	194.5	10.2	42.8

Superindices represent the number of unsolved instances using FMBV(G) among the five instances considered in each row, within the time limit of one hour

aforementioned instances with FMDS (G) is 3600 s, the average time of solving all the 140 instances would be 583.3 s. Again a reduction of more than one order of magnitude is reached with the decomposition.

Table 4 Results for the Minimum Degree Sum problem

$\|V\|$	Linked	v^*	Lower bound	Time (s)		
				FMDS(G)	Algorithm 2	
					Easy	All
40	2	7.8	7.8	0.2	0.0	0.0
60	3	11.0	10.2	0.4	0.0	0.0
80	2	25.4	21.4	0.8	0.2	0.2
120	3	37.6	35.4	1.4	0.4	0.4
200	2	113.0	98.8	8.4	2.6	2.6
240	2	150.4	135.6	8.6	2.4	2.4
280	2	179.8	159.8	11.6	3.8	3.8
300	3	176.2	157.8	28.2	2.4	2.4
320	2	213.4	190.2	49.4	6.2	6.2
360	3	226.6	200.2	50.2	5.4	5.4
360	2	254.8	227.8	25.4	5.2	5.2
400	2	279.8	252.2	56.2	13.4	13.4
420	3	270.4	236.6	^{1}61.5	3.7	8.2
480	3	321.6	286.0	95.8	8.0	8.0
500	2	381.8	348.4	26.2	8.6	8.6
540	3	369.6	326.2	^{1}37.7	8.7	19.4
600	3	422.8	385.0	331.2	12.4	12.4
600	2	483.8	440.2	213.0	22.6	22.6
700	2	572.6	518.0	268.6	31.0	31.0
750	3	571.4	520.6	^{2}91.0	10.6	18.6
800	2	690.6	637.0	^{1}643.5	48.0	54.4
900	3	728.2	670.4	467.4	29.2	29.2
900	2	785.4	718.6	^{1}979.5	59.25	75.4
1000	2	888.0	826.0	448.4	53.2	53.2
1050	3	870.0	791.2	^{3}204.5	12.5	70.0
1200	3	1030.6	946.6	^{1}1196.7	40.7	43.8
1350	3	1171.6	1082.8	^{3}1562.5	29.5	87.8
1500	3	1347.2	1250.6	^{1}1298.5	44.5	47.8
Averages		449.33	410.0	248.1	15.7	22.6

Superindices represent the number of unsolved instances using FMBV(G) among the five instances considered in each row, within the time limit of one hour

6 Conclusions

The relation between three different optimal spanning trees on connected graphs and their counterparts on the blocks of the graphs have been studied in the paper. In the case of the Minimum Leaves problem, simply joining the optimal trees of the blocks, the optimal

solution of the problem on the graph is obtained. Nevertheless, in the Minimum Branch Vertices and the Minimum Degree Sum problems, where only the nodes with degree greater than two contribute the objective function, the optimal solution cannot be obtained in this way. For these two objectives, ad-hoc decomposition methods and algorithms have been developed that allow to solve the problem by iteratively solving certain adapted subproblems on the blocks. The decomposition based on articulation vertices is a generalization of the decomposition approach proposed for the MBV in [15] in the sense that the graph of interest is split into subgraphs in order to reduce the computational effort. Our contribution is the design of a method to deal with articulation vertices with degree two (in the block-cutpoint graph), as well as the extension of the idea to other two related problems, the ML and the MDS. Each of the problems requires a different way of linking the partial solutions associated to the subgraphs produced by the decomposition, precisely due to the use of degree-two articulation vertices.

Instances of graphs that contain large blocks have been generated to check the efficiency of the algorithms. The same formulation (with slight modifications) has been used to solve both, the problem on the whole graph and the subproblems on the blocks. Despite the fact that the number of problems to be optimally solved is larger in the case of the algorithm, the computational results show the clear superiority of the decomposition.

We would like to close these conclusions mentioning that further advances in the resolution of the three problems we dealt with in this article could be perfectly combined with our decomposition method to produce even better results when the graph can be efficiently decomposed by using the articulation vertices. The great thing of our approach is that it can be used with any formulation or algorithm that can be devised to solve the subproblems. For example, in the recent paper [23] a very efficient branch-and-bound method for the MBV is described that shows to be able to produce very good computational results. Then, the novel ideas in this new approach could be incorporated to ours to try to produce even better results. We consider this combination an interesting matter for future research.

Acknowledgements This work has been supported by Ministerio de Economía y Competitividad, Projects MTM2015-65915-R and MTM2015-68097-P (MINECO/FEDER), Fundación Séneca, Project 19320/PI/14 and *Fundación BBVA*, Project "Cost-sensitive classification. A mathematical optimization approach" (COSECLA).

References

1. Bhatia, R., Khuller, S., Pless, R. Sussmann, Y.J.: The full degree spanning tree problem. In: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 864–865 (1999)
2. Carrabs, F., Cerulli, R., Gaudioso, M., Gentili, M.: Lower and upper bounds for the spanning tree with Minimum Branch Vertices. Comput. Optim. Appl. **56**, 405–438 (2013)
3. Cerrone, C., Cerulli, R., Raiconi, A.: Relations, models and a memetic approach for three degree-dependent spanning tree problems. Eur. J. Oper. Res. **232**, 442–453 (2014)
4. Cerulli, R., Gentili, M., Iossa, A.: Bounded-degree spanning tree problems: models and new algorithms. Comput. Optim. Appl. **42**, 353–370 (2009)
5. Chimani, M., Spoerhase, J.: Approximating spanning trees with few branches. Theory Comput. Syst. **56**, 181–196 (2015)
6. Contreras, I., Fernández, E., Marín, A.: Tight bounds from a path based formulation for the tree-of-hubs location problem. Comput. Oper. Res. **36**, 3117–3127 (2009)
7. Diestel, R.: Graph Theory, 3rd edn. Springer, New York (2005)
8. Fernandes, L., Gouveia, L.: Minimal spanning trees with a constraint on the number of leaves. Eur. J. Oper. Res. **104**, 250–261 (1998)

9. Fürer, M., Raghavachari, B.: An NC approximation algorithm for the minimum degree spanning tree problem. In: Proceedings of the 28th Annual Allerton Conference on Communication, Control and Computing, pp. 274–281 (1990)
10. Gargano, L., Hell, P., Stacho, L., Vaccaro, U.: Spanning trees with bounded number of branch vertices. In: Lecture Notes in Computer Science 2380, pp. 355–365. Springer, Berlin (2002)
11. Landete, M., Marín, A.: Looking for edge-equitable spanning trees. Comput. Oper. Res. **41**, 44–52 (2014)
12. Lu, H.I., Ravi, R.: The power of local optimization: approximation algorithms for maximum-leaf spanning tree. In: Proceedings of the Annual Allerton Conference on Communication Control and Computing, vol. 30, pp. 533–533. University of Illinois (1992)
13. Marín, A.: Exact and heuristic solutions for the minimum number of branch vertices spanning tree problem. Eur. J. Oper. Res. **245**, 680–689 (2015)
14. Matsuda, H., Ozeki, K., Yamashita, T.: Spanning trees with a bounded number of branch vertices in a claw-free graph. Graphs Combin. **30**, 429–437 (2014)
15. Melo, R.A., Samer, P., Urrutia, S.A.: An effective decomposition approach and heuristics to generate spanning trees with a small number of branch vertices. Comput. Optim. Appl. **65**, 821–844 (2016)
16. Merabet, M., Durand, S., Molnar, M.: Minimization of branching in the optical trees with constraints on the degree of nodes. In: ICN 2012, The Eleventh International Conference on Networks, pp. 235–240 (2012)
17. Narula, S.C., Ho, C.A.: Degree-constrained minimum spanning tree. Comput. Oper. Res. **7**, 239–249 (1980)
18. Öncan, T.: New formulations for the Minimum Branch Vertices Problem. In: Proceedings of the World Congress on Engineering and Computer Science, San Francisco, vol. II (2014)
19. Rossi, A., Singh, A., Shyam, S.: Cutting-plane-based algorithms for two branch vertices related spanning tree problems. Optim. Eng. **15**, 855–887 (2014)
20. Salamon, G., Wiener, G.: On finding spanning trees with few leaves. Inf. Process. Lett. **105**, 164–169 (2008)
21. Silva, D.M., Silva, R.M.A., Mateus, G.R., Gonçalves, J.F., Resende, M.G.C., Festa, P.: An iterative refinement algorithm for the Minimum Branch Vertices Problem. In: Lecture Notes in Computer Science 6630, pp. 421–433. Springer, Berlin (2011)
22. Silva, R.M.A., Silva, D.M., Resende, M.G.C., Mateus, G.R., Gonçalves, J.F., Festa, P.: An edge-swap heuristic for generating spanning trees with minimum number of branch vertices. Optim. Lett. **8**, 1225–1243 (2014)
23. Silvestri, S., Laporte, G., Cerulli, R.: A branch-and-cut algorithm for the Minimum Branch Vertices spanning tree problem. Comput. Oper. Res. **81**, 322–332 (2017)
24. Skiena, S.S.: The Algorithm Design Manual. Springer, New York (2008)
25. Sundar, S., Singh, A., Rossi, A.: New heuristics for two bounded-degree spanning tree problems. Inf. Sci. **195**, 226–240 (2012)