RAIRO-Oper. Res. 49 (2015) 143–160
DOI: 10.1051/ro/2014029

RAIRO Operations Research
www.rairo-ro.org

AN EXACT METHOD FOR SOLVING THE BI-OBJECTIVE MINIMUM DIAMETER-COST SPANNING TREE PROBLEM

Ernando Gomes de Sousa[1], Andréa Cynthia Santos[2] and Dario José Aloise[1]

Abstract. In this work, we propose a procedure to compute Pareto-optimal fronts for the bi-objective Minimum Diameter-Cost Spanning Tree problem (bi-MDCST). The bi-MDCST aims at finding spanning trees with minimum total cost and minimum diameter. Strategic decision problems for high-speed trains infrastructure, as well as tactical and operational optimization problems for network design and transportation can be modeled as bi-MDCST. The proposed exact procedure makes use of components from the multi-objective exact method Parallel Partitioning Method, and Pareto-optimal fronts have been computed for two benchmark instances from the literature. To the best of our knowledge, there are no works dedicated to providing Pareto-optimal fronts for the bi-MDCST.

Keywords. Spanning trees, multi-objective optimization, Parallel Partitioning Method.

Mathematics Subject Classification. 90-08, 90C27, 90C29, 68W99.

Received April 26, 2014. Accepted May 5, 2014.

[1] Universidade do Estado do Rio Grande do Norte, UERN Rua Almino Afonso, 478, CEP 59.610-210, Mossoró, RN, Brasil. eernandogomess@gmail.com;darioaloise@uern.br

[2] ICD-LOSI, Université de Technologie de Troyes 12, rue Marie Curie, CS 42060, 10004 Troyes Cedex, France. andrea.duhamel@utt.fr

Article published by EDP Sciences

1. Introduction

The bi-objective Minimum Diameter-Cost Spanning Tree problem (bi-MDCST) is defined in an undirected and connected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges with costs $c_{ij} \geq 0$ associated with each edge $[i, j] \in E$. By definition, a spanning tree $\mathcal{T}$ of G is a connected subgraph of G, without cycles and with $|V|$-1 edges. A Minimum weight Spanning Tree (MST) of G is a spanning tree with minimum total cost. Moreover, the diameter of a tree $\mathcal{T}$ is the number of edges in its longest path. Thus, a minimum diameter spanning tree is a spanning tree with the smallest diameter possible among all spanning trees $\mathcal{T}$ of G. Polynomial time algorithms are available to compute an MST such as Kruskal and Prim [5], as well as computing the diameter of a tree. However, the bi-MDCST is a NP-hard problem as shown in [16].

Strategic decision problems for high-speed trains infrastructure, as well as tactical and operational optimization problems for network design and transportation can be modeled as bi-MDCST. The minimum spanning tree cost usually refers to reducing the infrastructure costs, while the minimum diameter mostly stands for improving the quality of service. A problem strongly related to the bi-MDCST is the Bounded Diameter Minimum Spanning tree problem (BDMST) which seeks spanning trees where the path between all pairs of nodes has up to D edges, where $D \geq 2$ edges. The BDMST is NP-hard when $3 < D < |V| - 1$ [10]. Several works address the BDMST. For instance, exact algorithms are proposed by [12,22], heuristics and metaheuristics are found in [13, 19, 24], mathematical formulation and valid inequalities are presented in [1, 11, 27]. The bi-MDCST generalizes the BDMST since it minimizes the diameter and the cost simultaneously, and seeks the Pareto front taking both objectives. Whenever a unique diameter D is considered, the bi-MDCST is reduced to the BDMST.

Several works in the literature deal with bi-objective minimum spanning tree problems with two cost objective functions, such as heuristics [2, 33], enumeration methods [29], and Branch-and-Bound [28]. For such problems, two cost functions c_{ij} and f_{ij} are associated with each edge $[i, j] \in E$, and the problem relies on computing spanning trees such that costs c_{ij} and f_{ij} are minimized simultaneously. Thus, the mathematical formulations solve a simple MST, considering the two objectives. In the bi-MDCST, the diameter minimization is related to the number of edges in the tree between each pair of nodes. As a consequence, the mathematical formulations need to be adapted to take this fact into account.

Some works in the literature deal with the bi-MDCST. A theoretical study of approximative algorithms for the bi-MDCST and MST with degree constraints is introduced in [20]. Heuristics based on multi-objective genetic algorithms are proposed by [18, 25, 26]. Work [25] presents a standard Multi-Objective Genetic Algorithm (MOEA). Computational results are provided and compared with the following greedy heuristics: One Time Tree construction (OTT) [6] and Randomized Greedy Heuristic (RGH) [23]. The study [26] reproduces the MOEA proposed by [25] with some improvements, and proposes a Non-dominated Sorting Genetic

Algorithm (NSGA-II). Extensive results are presented to calibrate and compare such heuristics. Moreover, a comparison is made with optimal solutions found by a multiflow formulation, where the objectives are optimized in priority order, and with optimal results for the BDMST. A multiflow formulation is also proposed by [26], and results are given for an optimization in two phases, *i.e.*, each objective is minimized in turn and the diameter is considered as the priority objective.

In this study, we propose a procedure to compute Pareto-optimal fronts for the bi-MDCST based on the Parallel Partitioning Method. To the best of our knowledge, there are no works in the literature dedicated to finding the complete Pareto-optimal solutions for the bi-MDCST. Thus, one contribution of this work is to present the Pareto-optimal fronts for two important test sets, covering instances introduced by [26,27], and usually used in the works for the bi-MDCST. In the next section, definitions and notations used in this work are given. Then, some classic exact procedures for multi-objective optimization problems are briefly reviewed in Section 3. The proposed procedure is detailed in Section 4. Finally, computational results are presented in Section 5, followed by concluding remarks in Section 6.

2. Definitions and notations

A multi-objective optimization problem $\mathcal{P}$ can be modeled as $\{\min f(\vec{x}) | \vec{x} \in \mathcal{X}\}$, where $\mathcal{X}$ is the feasible solution space of $\mathcal{P}$, and $f(\vec{x})$ is the vector of objectives $(f_1(\vec{x}), f_2(\vec{x}), \ldots, f_n(\vec{x}))$ to be optimized. Multi-objective problems involve several compromise solutions among the focused objectives. Such a set of solutions is usually referred as the Pareto front which can be defined by means of the dominance concept given below. Without loss of generality, consider a minimization problem for which a solution x is said to dominate y if and only if, it satisfies the conditions given in equation (2.1).

$$\begin{cases} f_k(\vec{x}) \leq f_k(\vec{y}) & \forall k \in 1 \ldots n \quad \text{and} \\ f_k(\vec{x}) < f_k(\vec{y}) & \exists k \in 1 \ldots n \end{cases} \tag{2.1}$$

A Pareto solution $f_k(\vec{x^*})$ is said to be Pareto-optimal if no solution belonging to $\mathcal{X}$ dominates $f_k(\vec{x^*})$. The Pareto-optimal front is composed of the set of Pareto-optimal solutions (non-dominated).

The bi-MDCST is defined in a graph $G = (V, E)$, as mentioned before, and since it is a bi-objective problem $n = 2$. Let $\mathcal{T} = (V, E')$ be a spanning tree of G, with $E' \subseteq E$. A spanning tree $\mathcal{T}$ has a unique path ρ_{ij} between each pair of nodes $i, j \in V, i \neq j$. Denote by d_{ij} the number of edges in ρ_{ij}. The diameter D of $\mathcal{T}$ is defined as $D = \max\{d_{ij} : \forall i, j \in V, i \neq j\}$, *i.e.*, the number of edges in the longest path of $\mathcal{T}$. Moreover, let the first objective f_1 be the minimization of the total cost considering a spanning tree $\mathcal{T}$ of G, and let the second objective f_2 be the minimization of the diameter D, considering $\mathcal{T}$. The bi-MDCST seeks a

set of Pareto-optimal spanning trees $\mathcal{T}$ of G where f_1 and f_2 are simultaneously minimized.

Handler [15] defined a property to characterize and build spanning trees whether D is odd or even. Property 1 is used in the remainder of this work to obtain trees for some polynomial cases, and in the mathematical formulations.

Property 1. Whenever D is even, the spanning tree has a central vertex i such that no other vertex is more than $D/2$ edges away from i. If D is odd, the spanning tree has a central edge $e = [i, j]$, such that all vertices $k \in V \backslash \{i, j\}$ are no more than $(D-1)/2$ edges away from one extremity of e.

3. Multi-objective exact methods

Several exact methods can be applied to deal with multi-objective problems. Here, we briefly describe the ϵ-constraint, the Two-Phase Method (TPM), and the Parallel Partitioning Method (PPM), in particular for dealing with bi-objective problems, since some of their components are used in the proposed method. References [4,9,32,34] are interesting entry points for recent advances and bibliographical review on multi-objective strategies.

The ϵ-constraint is based on the ϵ-constraint enumeration strategy proposed by [14]. The general idea is to optimize one objective and consider the second objective as an additional constraint of the problem. Thus, the constrained optimization problem $\mathcal{P}'$ is defined as $\{\min\ f_1(\vec{x}) | f_2(\vec{x}) \leq \epsilon; \vec{x} \in \mathcal{X}\}$. The ϵ-constraint uses the optimal value of $f_1(\vec{x})$ as a bound on the search space. As a consequence, the search space is constrained by successively optimizing the objective $f_1(x)$ and considering the new solution value as a new bound. The procedure stops when no solution can be found. A successful application is found in [3] for the Traveling Salesman Problem with profits.

The TPM has been proposed by [30] and it is composed of two phases which are briefly described in the sequence. Applications of the TPM is found in [30,31], respectively for the bi-objective assignment problem and for the bi-objective 0-1 knapsack problem. In the first phase of the TPM, two non-dominated solutions are computed, which can be done, for example, by independently optimizing each objective $f_1(x)$ and $f_2(x)$. Then, non-dominated solutions are computed using the aggregating strategy and two target solutions. The aggregating strategy works as follows: considering a bi-objective problem, a linear combination $\alpha f_1(x) + (1 - \alpha) f_2(x)$, where $\alpha \in [0, 1]$ is considered. Whenever $\alpha = 1$, the first objective $f_1(x)$ is optimized, on the contrary if $\alpha = 0$, the second objective $f_2(x)$ is optimized. The procedure is recursively repeated to computed new solutions. When the procedure is not able to find new non-dominated solutions, it moves to the second phase, where a search is done between each pair of solutions found in the first phase.

The PPM has three distinct phases and it was developed to improve the second phase of the TPM. In particular, the search space between each pair of solutions

becomes identical, *i.e.*, the search space is partitioned using one of the objectives and such that the partitions have identical size. A comparison between the TPM and the PPM applied to the bi-objective permutation flowshop problem is done in [17]. Results show the superiority of the PPM. In the first phase of the PPM, upper and lower bounds in the search space are computed by doing an optimization of each objective $f_1(x)$ and $f_2(x)$. In the second phase, the search space is partitioned considering the bounds found in the first phase, and one of the objective function. The choice of the objective function used to define each partition is an important issue, which strongly depends on the problem and the objectives. In the following, the procedure generates a solution in each partition, using the ϵ-constraint strategy. The constrained problem $\mathcal{P}'$ is considered at this stage as a set of optimization problems with a unique objective, one for each partition that are recursively solved. Several strategies can be considered to solve at optimality these sub-problems such as Branch-and-Bound, linear and non-linear solvers. The third phase seeks solutions into the space between each pair of neighbour solutions found in the previous phases. The PPM has been extended by [8] to deal with multiple objectives, called of K-PPM method.

The first and the second phases of the PPM are based on the ϵ-constraint, it differs from the ϵ-constraint since the search space is partitioned, and upper and lower bounds are considered in the PPM instead of just an upper bound as in the ϵ-constraint method. The third phase of the PPM uses ideas from the TPM to fill the space among neighbour solutions. The search of new solutions is done using a linear combination of the objectives, taking as a target the rectangle between each pair of neighbour solutions. The PPM improves the TPM since the search space between each pair of solutions has a similar size.

4. An exact method for the bi-MDCST

In this section, the proposed exact procedure for the bi-MDCST is presented in detail, followed by a description of the mathematical formulations and an algorithm. The method makes use of components from the PPM.

Initially, the proposed procedure applies the first phase of the PPM method. Then, lower and upper bounds are computed, which correspond to the MST cost of G and to the diameter of G. The first bound on the search space is obtained by applying Prim's algorithm [5] to compute an MST of G. The MST cost is considered as an upper bound on the search space, and its diameter is used as a target value to constraint the search space. Whenever a graph G has edges with similar costs, there are different MST of G. Figure 1 illustrates this situation, where the graph in Figure 1a has more than one MST. Two different MST are depicted in Figures 1b and 1c, with costs equal to 20 and the diameters are respectively equal to $D = 5$ (path between nodes 0 and 5) and $D = 6$ (path between nodes 1 and 6). Thus, the solution for the bi-MDCST in Figure 1c is dominated by the solution presented in Figure 1b due to the diameter value.

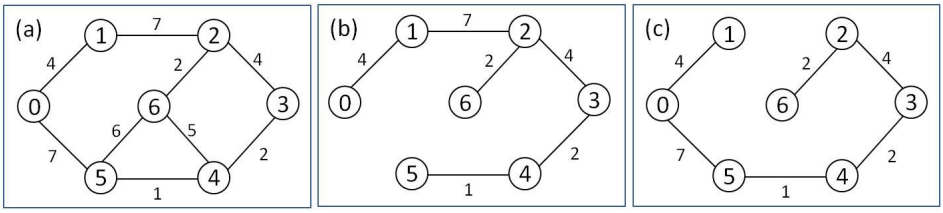

FIGURE 1. Graph with multiple MST.

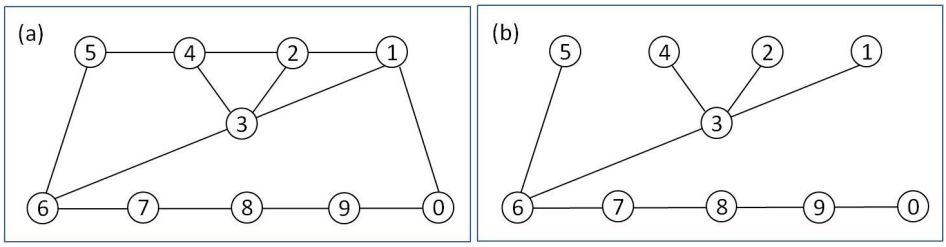

FIGURE 2. Sparse graph with $D = 4$ and a minimum diameter spanning tree with $D = 6$.

In a spanning tree, there is a unique path between each pair of nodes, while in a graph, several paths can exist among pairs of nodes. Thus, the diameter of a graph is computed as follows: let d'_{ij}, $i, j \in V$, be the minimum number of edges among every possible path between i and j in G, the diameter D' of G is equal to $D' = \max\{d'_{ij} : \forall i, j \in V, i \neq j\}$. It is obvious that the minimum diameter spanning tree of G has a diameter greater than or equal to the diameter of G. The second bound on the bi-MDCST search space is considered as the diameter of G. It important to highlight that, for some graphs, a spanning tree $\mathcal{T}$ of G with the diameter equals the diameter of G does not exist, as the example shown in Figure 2. A sparse graph with $D = 4$ is given in Figure 2a, and a spanning tree with $D = 6$ is illustrated in Figure 2b. One may note that no spanning tree can be built for this graph with $D = 4$ or $D = 5$.

Whenever a graph G has a diameter equal to $D = 2$ or $D = 3$ (a complete graph always has spanning trees of diameter $D = 2$ and $D = 3$), the MST with such diameter can be computed in polynomial time as shown in [26]. Whenever there is a spanning tree with $D = 2$, a solution with $D = 2$ can be computed in polynomial running time by using results from Property 1 (see Sect. 2). At each iteration, a spanning tree is built, whenever possible, considering each node $i \in V$ as the central node and connecting all other vertices $j \in V \backslash \{i\}$ to the tree using edge $[i, j] \in E$. Then, it remains to compare up to $O(|V|)$ spanning trees of G. This procedure has computational complexity $O(|V|^2)$. It is important to mention

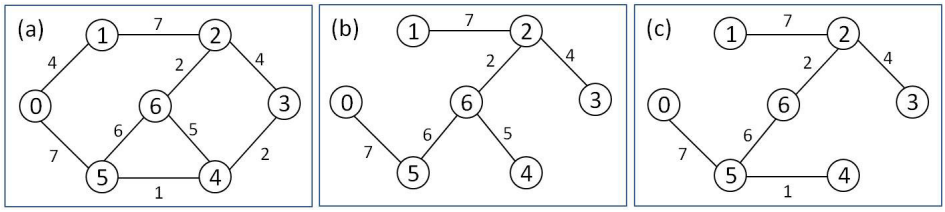

FIGURE 3. Example of a graph with different MDST.

that a central node can lead to an infeasible spanning tree in sparse graphs. In this case, the solution is discarded.

If a spanning tree of $D = 3$ exists, and following Property 1, a procedure builds a spanning tree at each iteration considering an edge $[i, j] \in E$ at a time as the central edge. Given a central edge $[i, j] \in E$, all vertices $k \in V \backslash \{i, j\}$ are connected to one of the central edge extremities i or j. The edge $[k, i]$ or $[k, j]$ with smallest cost enters the solution. Thus, up to $O(|E|)$ spanning trees are compared. As a consequence, computing a spanning tree of $D = 3$ takes $O(|E| \cdot |V|)$ in the worst case. When the diameter is $D \geq 4$, the problem becomes NP-hard and the mathematical formulations described in Section 4.1 are employed to find such solutions by means of a Mixed Integer Linear Programming (MILP) solver.

A graph G can have several Minimum Diameter Spanning Trees (MDST). An example is shown in Figure 3. A graph G is given in Figure 3a and two minimum diameter spanning trees are presented in Figures 3b and 3c, with diameter $D = 4$ and costs equal to 31 and 27, respectively. Thus, solution in Figure 3c dominates the solution depicted in Figure 3b because of the spanning tree cost.

The second step of the proposed procedure consists of partitioning the search space into identical rectangles. To accomplish that, it is necessary to decide which objective is taken as reference to decompose the search space. For the bi-MDCST, partitions are done considering the diameter since it corresponds to integer values and each rectangle's height is equal to 1. The search space is then decomposed in such a way that each partition contains a target diameter value.

After the decomposition, the procedure seeks an optimized solution in each partition, if it exists. This step is based on the ϵ-constraint method since the objective function dealing with the diameter is set as a constraint to the problem. As mentioned above, finding an MST with $3 < D < |V| - 1$ relies on a NP-hard problem. One may note that at this point the problem in each partition is a BDMST. Thus, proving optimality is limited by the instance size. Even so, it is interesting to analyse the Pareto-optimal front for some instances applied to the bi-MDCST and BDMST in the literature. Here, solutions with diameters $3 < D < |V| - 1$ are recursively solved for each partition using a MILP solver.

The mathematical formulations from [1], which were strengthened by [27], are used to solve the problems in each partition (see Sect. 4.1). In spite of the duality gap, these formulations are able to prove optimality for graphs with a higher

number of nodes and edges than the multiflow formulations proposed by [26]. A possible explanation is that the number of variables and constraints for the multiflow formulations are equal to $O(|V^4|)$, while in the models proposed by [1], they are significantly smaller (see Sect. 4.1).

It is worth mentioning that the formulations seek MST with $D \leq d$, where d is a positive integer. Thus, whenever the solution found has $D < d$, it means that all solutions from the target diameter d until the obtained diameter D are dominated. As a consequence, the proposed procedure does not need to run the formulations for all diameters, thus saving some running time.

The procedure performs the steps mentioned above and stops when it attains the diameter lower bound. The third phase of the PPM does not apply to the bi-MDCST since the diameter is a positive integer value. Thus, there are no solutions among neighbouring partitions.

4.1. Mathematical formulations

The MILP formulations considered are based on the Property 1 (Sect. 2), one for the case when D is odd and another for the case when D is even, referred in the sequence as D even case and D odd case.

An artificial node r is introduced in G with costs $c_{ri} = 0 \, \forall i \in V$, and corresponds to the source of commodities. Moreover, the formulations work on a directed graph, which allow us to control the diameter of the tree from the artificial node r to each node in the solution. Thus, a new graph G' is obtained from G to take into account the artificial node r, and to transform edges of G in arcs as follows: for each edge $[i, j] \in E$, with $i < j$, two arcs (i, j) and (j, i) are introduced in G', with costs $c_{ij} = c_{ji}$. Then, $G' = (V', A')$, where the set of vertices V' and the set of arcs A' are respectively given by $V' = V \cup \{r\}$ and $A' = A \cup \{(r, 1), \ldots, (r, |V|)\}$. After building a directed spanning tree, the arcs orientation and the artificial node are taken out of the solution.

The formulation for the D even case is presented from (4.1) to (4.6). The binary variables x_{ij} determine if an arc (i, j) is selected ($x_{ij} = 1$), or not ($x_{ij} = 0$) to enter the solution. Furthermore, variables u_i are associated with each vertex $i \in V'$. These variables specify the order each node is visited, considering for the artificial node, variable $u_r = 0$. Thus, the central vertex l of the solution is set with $u_l = 1$. Moreover, let $L = D/2$ for the D even case, following Property 1. It means that a feasible solution for the D even case has up to L edges away from r.

The objective function to minimize costs is presented in (4.1). Equation (4.2) ensures the artificial vertex r is connected to exactly one vertex $i \in V$. Constraints (4.3) define that a unique arc is incident to each vertex $i \in V$. Inequalities (4.4) correspond to an adaptation of the classic restrictions proposed by Miller, Tucker, and Zemlin (MTZ) [21] to determine the topological order. Here, the strengthened version of such constraints is applied (lifted) [7]. Concerning the diameter constraints, they are established as a consequence of

restrictions (4.4) and (4.6). Variables are defined from (4.5) to (4.6). This formulation has $O(|V'| + |A'|)$ variables and $O(|V| + |A'|)$ constraints.

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{4.1}$$

$$\sum_{j\in V} x_{rj} = 1 \tag{4.2}$$

$$\sum_{(i,j)\in A'} x_{ij} = 1 \qquad \forall j \in V \tag{4.3}$$

$$u_i - u_j + (L+1)x_{ij} + (L-1)x_{ji} \leq L \qquad \forall (i,j) \in A' \tag{4.4}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A' \tag{4.5}$$

$$0 \leq u_i \leq L+1 \qquad \forall i \in V' \tag{4.6}$$

The formulation for the D odd case is presented from (4.7) to (4.15) and uses variables x_{ij} and u_i as for the D even case. In addition, binary variables $z_{ij} \forall [i,j] \in E$, $i < j$, are responsible for selecting the central edge. Whenever $z_{ij} = 1$, $[i,j]$ is taken as the central edge, otherwise $z_{ij} = 0$. In this formulation, the artificial vertex r has to be connected to exactly two vertices $p, q \in V$ in the final solution. This way, the central edge $[p,q] \in E$ is implicitly selected. After getting the final solution, the arc orientation, the artificial node and arcs $[r,p]$ and $[r,q]$ are taken away from the resulting spanning tree $\mathcal{T}^*$, and edge $[p,q]$ enters $\mathcal{T}^*$.

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{[i,j]\in E} c_{ij}z_{ij} \tag{4.7}$$

$$\sum_{j\in V} x_{rj} = 2 \tag{4.8}$$

$$\sum_{(i,j)\in A'} x_{ij} = 1 \qquad \forall j \in V \tag{4.9}$$

$$\sum_{[i,j]\in E} z_{ij} = 1 \tag{4.10}$$

$$z_{ij} = x_{ri} \cdot x_{rj} \qquad \forall [i,j] \in E \tag{4.11}$$

$$u_i - u_j + (L+1)x_{ij} + (L-1)x_{ji} \leq L \qquad \forall (i,j) \in A' \tag{4.12}$$

$$0 \leq u_i \leq L+1 \qquad \forall i \in V' \tag{4.13}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A' \tag{4.14}$$

$$z_{ij} \in \{0,1\} \qquad \forall [i,j] \in E \tag{4.15}$$

The objective function (4.7) has a new term $\sum_{[i,j]\in E} c_{ij}z_{ij}$ to compute the central edge cost. Equation (4.8) establishes that the artificial vertex r is connected

```
   Input: G = (V, E)
 1 k ← 1;
 2 T_k ← Compute_MST;
 3 F* ← F* ∪ {T_k};
 4 UB ← Get_diameter (T_k);
 5 LB ← Get_diameter (G);
 6 k ← k + 1;
 7 d ← UB;
 8 if (UB ≠ LB) then
 9     while (d ≥ LB) do
10         if (d > 3) then
11             T_k ← Run_MILP_solver (G, d);
12             if (T_k ≠ ∅) then
13                 F* ← F* ∪ {T_k};
14                 if (Get_diameter (T_k) < d) then
15                     d ← Get_diameter (T_k);
16                 end
17                 k ← k + 1;
18             end
19         else
20             T_k ← Run_polynomial_procedure (G, d);
21             if (T_k ≠ ∅) then
22                 F* ← F* ∪ {T_k};
23                 k ← k + 1;
24             end
25         end
26         Cleaning(T_k, F*);
27         d ← d − 1;
28     end
29 end
30 return F*;
```

Algorithm 1: An exact procedure for solving the bi-MDCST.

to exactly two vertices of V. Constraints (4.9) state that each arc is incident to each vertex $i \in V\backslash\{r\}$. Equation (4.10) ensures that there is a unique central edge in the final solution. Inequalities (4.11) are non-linear and link the decision variables z_{ij} to x_{ij}. These restrictions state that the artificial vertex is connected to exactly two vertices $i \in V$, and are easily linearized as shown in [27]. The lifted MTZ constraints are defined in (4.12). Finally, the domain of variables is given from (4.13) to (4.15). This formulation has $O(|V'|+|A'|)$ variables and $O(|V|+|A'|)$ constraints.

4.2. An algorithm for the proposed exact method

A pseudo-code for the proposed exact method is given in Algorithm 1, for which a graph $G = (V, E)$ is the input parameter. Let UB, LB, $\mathcal{F}^*$, and d be respectively

the upper bound for the cost, the lower bound for the diameter, the set of Pareto-optimal solutions, and an auxiliary variable to determine the target diameter at each partition. Moreover, $\mathcal{T}_k$ is the kth solution in the Pareto-optimal front.

Lines 1 to 7 are the initialization steps. Variable k is set in line 1. Then, an MST of G is computed in line 2 using Prim's algorithm. The solution $\mathcal{T}_k$ is set in the Pareto-optimal front $\mathcal{F}^*$ in line 3. After that, Variables, UB, LB, k and d are respectively updated in lines 4 to 7. The diameter of a graph G is obtained by a trivial adaptation of a breadth-search first algorithm (line 6). The procedure checks if there is more than one solution in the search space in line 8. If so, loop 9 to 28 is repeated for each partition until attaining the lower bound. Whenever $d > 3$ (line 10), the MILP is applied (line 11). If a solution exists for the corresponding diameter, $\mathcal{F}^*$ is updated in line 13. The mathematical models seek MST with the diameter less than or equal to the target value d. Thus, the goal of lines 14 to 16 is to check if a solution is found for the target diameter or not. If there is no solution with the target diameter d (line 14), d is updated (line 15). Lines 20 to 25 deal with the polynomial cases ($d = 2$ or $d = 3$) and perform the polynomial algorithms (line 20) mentioned in Section 4. Whenever a solution is found (line 21), the Pareto front $\mathcal{F}^*$ and k are respectively refreshed in lines 22 and 23. Finally, the Cleaning procedure determines whenever the new solution $\mathcal{T}_k$ dominates solutions belonging to the incumbent Pareto-optimal front $\mathcal{F}^*$ (line 26), and the target value d is refreshed (line 27). The procedure stops after inspecting all possible target diameters between the lower and upper bounds, and the Pareto-optimal front is returned (line 30). The search space is not explicitly decomposed. In fact, the procedure computes solutions between the lower and upper bounds, by considering a target diameter in each partition.

5. Computational results

The computational experiments were carried out on an Intel Core i5 with 2.53 GHz clock and 4 GB of RAM. The proposed procedure has been developed in C++, using the Dev-C++ 4.9.9.2. The code is coupled to MILP solver CPLEX 12.0 using default parameters.

A number of 20 instances from [27] and 22 proposed by [26] are used in the computational experiments, respectively referred to the first test set and the second test set. In the naming format for instances in the first set, $k\_Vy\_Az\_Dw$ corresponds to: k identifies if the graph is complete ("C") or sparse ("S"), Vy stands for the number of vertices y, Az specifies the number z of edges belonging to G, and Dw depicts the diameter w tested by [27]. For the second test set, the format c_Vy_Z stands for: "c" indicates if it is the *Hamiltonien cycle* instances (the connectivity is ensured by an arbitrary *Hamiltonien cycle*) or "p", the *Hamiltonien path* instances (the connectivity is ensured by an arbitrary *Hamiltonien path*), Vy indicates the number of vertices y, Z the graph density, which determines the number of additional edges to be included after ensuring connectivity. Thus,

for the *Hamiltonien cycle* instances, density is set to {0.2, 0.3, 0.4}, while the *Hamiltonien path* instances, density is set to {0.1, 0.09, 0.08}.

Results are presented in Tables 1 and 2, respectively for instances from works [26, 27]. Each line corresponds to an instance, and the first column refers to the instance names. Some instances treated by [27] differ only by the diameter, the graph G is identical such as the instance with 15 vertices. Here, only instances of distinct graphs are considered. Column "Q" shows the total number of solutions found in the Pareto-optimal front. For each diameter from columns 3 to 21, the MST cost is provided if the corresponding solution is non-dominated. Whenever it is dominated the symbol "–" is depicted. Moreover, if a graph does not allow a solution with a specific diameter, the symbol "*" are used. An empty cell means the corresponding diameter is not focused for the corresponding instance. Finally, the running time in seconds to compute the complete Pareto-optimal front is given in the column $time(s)$.

Results in Table 1 show that except for the instance $S\_V40\_A100\_D4$, the Pareto-optimal front has at most 14 solutions (14 target diameters). In terms of running time, finding the complete Pareto-optimal front for this test set consumes about 50 seconds for the instances in complete graphs with 10 and 15 nodes, and sparse graphs with 20 nodes; less than 40 minutes for instances with 20 nodes in complete graphs and about 13 hours for the instances with 40 nodes. For this test set, the instances with diameter equal to $D = 4$ and $D = 5$ take the biggest part of the running time. The problem often becomes easy when the diameter is close to the MST diameter. Moreover, instances with identical costs seems to be more difficult to solve. A possible explanation is that they impose more combinatorial choices in the Branch-and-Bound tree, as is the case of instance $S\_V40\_A100\_D4$.

Figures 4 illustrates the complete Pareto-optimal front for instances from the first test set. The results for other instances in this test set behave similarly. However, we publish the solutions and the Pareto-optimal fronts for the two test sets on the site `http://di.uern.br/dario/bi-mdcst-problem/`. Some Pareto-optimal fronts appear with dominated solutions such as the instance $C\_V20\_A190\_D4$ with $D = 11$ and $D = 13$. Moreover, some instances have the MST dominated, as is the case of the instance $C\_V20\_A190\_D5$, where the MST diameter is $D = 13$, and this solution is dominated by the solution with $D = 12$. Finally, solutions with $D = 2$ and $D = 3$ have, in most of the cases, a higher cost value when compared with the other diameters. This generates a disruption in the Pareto-optimal front.

Table 2 presents results for the sparse graph instances for the second test set. The running times are smaller than those to solve the first test set. The number of partitions to be inspected are up to 9 and up to 15, respectively for the *Hamiltonien cycle* and *Hamiltonien path* instances. Due to the the fact that the graphs are sparse, it is not surprising that some solutions with specific diameters do not exist. For such a test set, the largest number of partitions without solutions varies from 1 to 7, mostly they are close to the lower bound. This indicates that

TABLE 1. Pareto-optimal front results for the bi-MDCST.

		Diameters																		
Instances	Q	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	time(s)
C_V10_A45_D4	8	392	279	252	234	218	210	208	206												0.70
C_V10_A45_D5	5	382	330	247	230	218															0.31
C_V10_A45_D6	5	360	275	235	226	221															0.26
C_V10_A45_D7	8	447	292	250	226	211	203	199	198												0.72
C_V10_A45_D8	6	389	297	243	236	233	232														0.30
C_V10_A45_D9	7	483	347	284	270	263	256	254													0.55
C_V15_A105_D4	11	681	482	346	331	314	303	295	290	286	–	281	277								25.16
C_V15_A105_D8	7	491	390	291	274	256	247	240													23.81
C_V20_A190_D4	11	629	481	349	330	302	287	282	276	274	–	272	–	271	–						520.53
C_V20_A190_D5	11	792	641	441	414	382	367	352	350	347	343	341	–	–							199.01
C_V20_A190_D6	14	681	498	343	316	298	287	278	276	272	269	267	265	263	261						387.10
C_V20_A190_D7	8	716	546	414	376	345	333	321	316											–	248.32
C_V20_A190_D8	13	789	592	421	382	353	344	331	323	317	311	304	299	295							2241.32
C_V20_A190_D9	12	822	599	418	390	356	345	333	327	324	–	–	321	316	313						1014.45
S_V20_A50_D4	9	693	601	442	410	369	357	340	334	–	332										11.47
S_V20_A50_D5	9	807	694	417	381	347	330	322	318	315											14.15
S_V20_A50_D6	10	755	617	395	366	329	323	321	309	303	300										36.35
S_V20_A50_D7	9	707	598	404	386	370	362	358	357	356											22.86
S_V20_A50_D8	10	822	679	434	411	379	373	366	362	359	357										49.60
S_V40_A100_D4	19	1574	1403	755	706	615	589	567	552	546	538	528	525	523	508	505	503	501	500	–	46 690.46

TABLE 2. Pareto-optimal front results for the bi-MDCST.

		Diameters																	
Instances	Q	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	time(s)
c_v10_0.3	2			*	*	434	420														0.13
c_v10_0.4	3			305	275	269															0.09
c_v15_0.2	4			*	*	*	497	453	*	451	443										0.54
c_v15_0.3	7			635	537	469	435	428	418	415											1.87
c_v15_0.4	6			465	407	391	378	376	375	–											1.37
c_v20_0.2	6			*	980	844	813	794	787	776											2.68
c_v20_0.3	5			1001	618	521	478	470													19.38
c_v20_0.4	7			614	536	493	480	474	468	467											13.92
c_v25_0.2	8			*	940	741	682	641	624	611	606	599									61.23
c_v25_0.3	5			731	648	556	535	–	529												18.08
c_v25_0.4	8			569	519	481	471	461	455	451	–	447									18.08
p_v25_0.09	2									*	*	*	*	*	*	*	1146	1144			1.47
p_v25_0.10	7							*	1323	1259	1223	1215	1190	–	1187	1186					2.05
p_v30_0.08	5								*	*	*	1858	1847	–	1844	–	1841	–	–	1838	3.34
p_v30_0.09	6						*	*	1407	1276	1250	1237	1234	1217							8.41
p_v30_0.10	8						*	*	1315	1249	1227	1189	1180	1168	1160	–	1140				39.8
p_v35_0.08	7							*	*	1677	1653	1635	–	1617	1611	–	–	–	1608	1604	23.4
p_v35_0.09	3					*	*	1702	1568	1513											47.77
p_v35_0.10	7						*	1243	1179	1166	1146	1135	1132	1112							17.52
p_v40_0.08	6							*	1889	1646	1640	1591	1566	1553							697.58
p_v40_0.09	6					*	1171	1613	1520	1471	1468	–	1467								484.66
p_v40_0.10	13				*	*	1651	1446	1373	1345	1337	1330	1329	1325	1324	1321	1317	1314	1313		1787.55

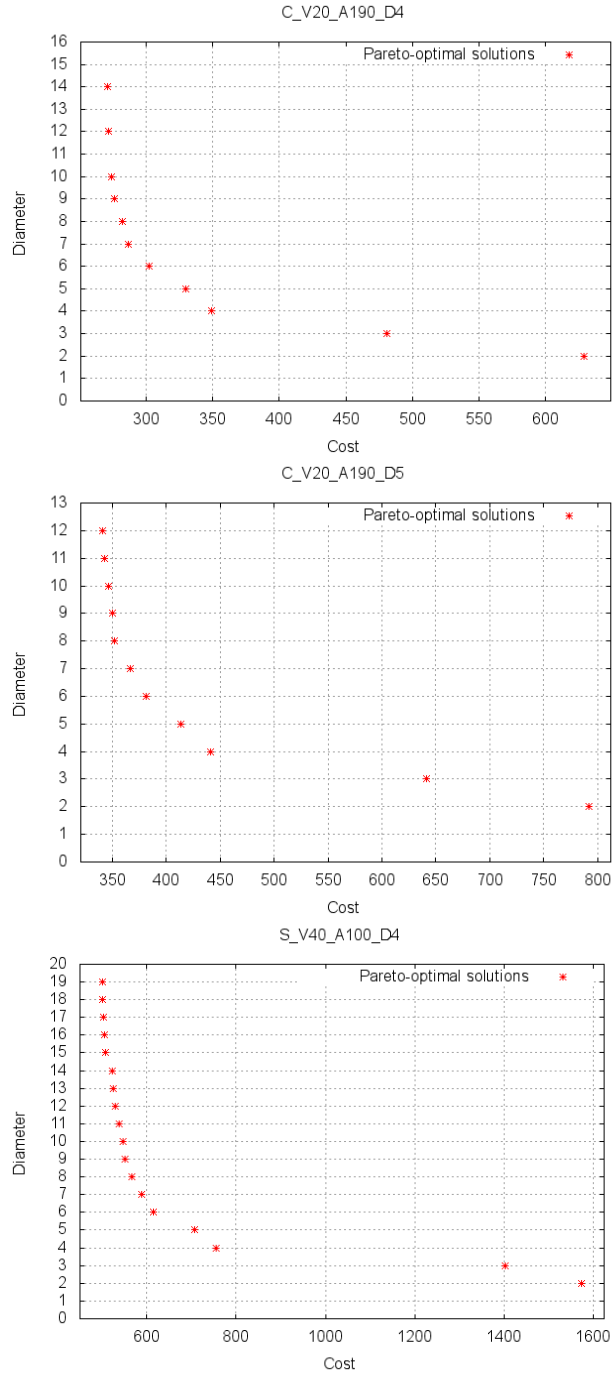

FIGURE 4. Pareto-optimal front for instances in the first test set.

for the second test set, the lower bound is not accurate. Thus, there is room for improving the lower bound computation.

6. Concluding remarks

In this work, an exact procedure to compute Pareto-optimal solutions for the bi-MDCST is proposed. It uses components from the PPM method. Moreover, the sub-problems in each partition are solved by MILP formulations from [27].

The sub-problems in each partition are NP-hard for some diameters since they correspond to the BDMST. In spite of that, the Pareto-optimal fronts have been computed for instances from two test sets, which is a significant contribution. In particular, these results can be used to analyse the quality of heuristic results for the bi-MDCST.

This study makes two other contributions involving the BDMST. First, a generalization of this problem is investigated, and second the problem is treated in the perspective of the multi-objective optimization. The concepts of dominance and Pareto front are applied. Thus, the results give information about the search space, for example: some diameters are not interesting because the solutions are dominated. Furthermore, such analysis can be also useful for decision makers interested on BDMST and bi-MDCST applications.

As future work, other methods to solve the sub-problems in each partition of the proposed method can be investigated such as *Branch-and-Cut*, column generation and enumeration algorithms. Moreover, heuristics and matheuristics can be developed to reduce the running time to obtain Pareto-optimal solutions. We are currently investigating a procedure which encloses cuts into the Branch-and-Bound tree in order to automatically manage the dominance concept.

References

[1] N.R. Achuthan, L.Caccetta, P.A. Caccetta and J.F. Geelen, Computational methods for the diameter restricted minimum weight spanning tree problem. *Austral. J. Combin.* **10** (1994) 51–71.

[2] J.E.C. Arroyo, P.S. Vieira and D.S. Vianna, A GRASP algorithm for the multi-criteria minimum spanning tree problem. *Ann. Oper. Res.* **159** (2008) 125–133.

[3] J.-F. Bérubé, M. Gendreau and J.-Y. Potvin, An exact ϵ-constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with profits. *Eur. J. Oper. Res.* **194** (2009) 39–50.

[4] A. Chinchuluun and P.M. Pardalos, A survey of recent developments in multiobjective optimization. *Ann. Oper. Res.* **154** (2007) 29–50.

[5] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to algorithms*. McGraw-Hill, New York (1990).

[6] N. Deo and A. Abdalla, Computing a diameter-constrained minimum spanning tree in parallel. *Lect. Notes Comput. Sci.* **1767** (2000) 17–31.

[7] M. Desrochers and G. Laporte, Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Oper. Res. Lett.* **10** (1991) 27–36.

[8] C.D. Dhaenems, J. Lemestre and E.G. Talbi, K-PPM: A new exact method to solve multi-objective combinatorial optimization problems. *Eur. J. Oper. Res.* **200** (2010) 45–53.
[9] M. Ehrgott and X. Gandibleux, A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum* **22** (2000) 425–460.
[10] M.R. Garey and D.S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman, New York (1979).
[11] L. Gouveia and T.L. Magnanti, Network flow models for designing diameter-constrained minimum-spanning and Steiner trees. *Networks* **41** (2003) 159–173.
[12] L. Gouveia, L. Simonetti and E. Uchoa, Modeling hop-constrained and diameter-constrained minimum spanning tree problems as Steiner tree problems over layered graphs. *Math. Program.* **128** (2011) 123–148.
[13] M. Gruber and G.R. Raidl, Variable neighborhood search for the bounded diameter minimum spanning tree problem. In P. Hansen, N. Mladenovic, J.A.M. Pérez, B.M. Batista, and J.M. MorenoVega, editors, *Proc. of the 18th Mini Euro Conference on Variable Neighborhood Search*, pages 1–11, Tenerife, 2005.
[14] Y. Haimes, L. Ladson and D. Wismer, On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Trans. Syst. Man Cybern. SMC* **1** (1971) 296–297.
[15] G.Y. Handler, Minimax location of a facility in an undirected graph. *Transp. Sci.* **7** (1978) 287–293.
[16] J.-M. Ho, D.T. Lee, C.-H. Chang and K. Wong, Minimum diameter spanning trees and related problems. *SIAM J. Comput.* **20** (1991) 987–997.
[17] J. Lemesre, C. Dhaenens and E.G. Talbi, Parallel Partitioning Method (PPM): A new exact method to solve bi-objective problems. *Comput. Oper. Res.* **34** (2007) 2450–2462.
[18] D.R. Lima, A.C. Santos and D.J. Aloise, Um algoritmo evolucionário NSGA-II para resolver o problema bi-objetivo da árvore geradora de custo e diâmetro mínimos. In *XVI Congreso Latino-Iberoamericano de Investigación Operativa, XLIV Simpósio Brasileiro de Pesquisa Operacional (CLAIO-SBPO)*, pages 689–699, 2012.
[19] A. Lucena, C. Ribeiro and A.C. Santos, A hybrid heuristic for the diameter constrained minimum spanning tree problem. *J. Glob. Optim.* **46** (2010) 363–381.
[20] M.V. Marathe, R. Ravi, R. Sundaram, S.S. Ravi, D.J. Rosenkrantz and H.B. HuntIII, Bicriteria network design problems. *J. Algor.* **28** (1998) 142–171.
[21] C.E. Miller, A.W. Tucker and R.A. Zemlin, Integer programming formulations and Traveling Salesman Problems. *J. ACM* **7** (1960) 326–329.
[22] T.F. Noronha, C.C. Ribeiro and A.C. Santos, Solving diameter-constrained minimum spanning tree problems by constraint programming. *Int. Trans. Oper. Res.* **17** (2010) 653–665.
[23] G.R. Raidl and B.A. Julstrom, Greedy heuristics and an evolutionary algorithm for the bounded-diameter minimum spanning tree problem. In *Proc. of the 18th ACM Symposium on Applied Computing*, pages 747–752, Melbourne (USA), 2003.
[24] C. Requejo and E. Santos, Greedy heuristics for the diameter-constrained minimum spanning tree problem. *J. Math. Sci.* **161** (2009) 930–943.
[25] S. Saha and R. Kumar, Bounded-diameter MST instances with hybridization of multi-objective EA. *Int. J. Comput. Appl.* **18** (2011) 17–25.
[26] A.C. Santos, D.R. Lima and D.J. Aloise, Modeling and solving the bi-objective minimum diameter-cost spanning tree problem. *J. Glob. Optim.* (2013) 1–22.
[27] A.C. Santos, A. Lucena and C.C. Ribeiro, Solving diameter constrained minimum spanning tree problem in dense graphs. *Lect. Notes Comput. Sci.* **3059** (2004) 458–467.
[28] F. Sourd and O. Spanjaard, A multiobjective Branch-and-Bound framework: Application to the biobjective spanning tree problem. *INFORMS J. Comput.* **20** (2008) 472–484.
[29] S. Steiner and T. Radzik, Computing all efficient solutions of the biobjective minimum spanning tree problem. *Comput. Oper. Res.* **35** (2008) 198–211.
[30] E.L. Ulungu and J. Teghem, The two Phases Method: An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences* **20** (1995) 149–165.

[31] M. Visée, J. Teghem, M. Pirlot and E.L. Ulungu, Two-Phases Method and Branch and Bound procedures to solve the bi-objective knapsack problem. *J. Glob. Optim.* **12** (1998) 139–155.
[32] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P.N. Suganthan and Q. Zhang, Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation* **1** (2011) 32–49.
[33] G. Zhou and M. Gen, Genetic algorithm approach on multi-criteria minimum spanning tree problem. *Eur. J. Oper. Res.* **114** (1999) 141–152.
[34] C. Zopounidis and P.M. Pardalos, *Handbook of multicriteria analysis*, vol. 103. Springer (2010).