J Glob Optim (2014) 60:195–216
DOI 10.1007/s10898-013-0124-4

Modeling and solving the bi-objective minimum diameter-cost spanning tree problem

Andréa Cynthia Santos · Diego Rocha Lima · Dario José Aloise

Received: 4 January 2013 / Accepted: 20 November 2013 / Published online: 6 December 2013

Abstract The bi-objective minimum diameter-cost spanning tree problem (bi-MDCST) seeks spanning trees with minimum total cost and minimum diameter. The bi-objective version generalizes the well-known bounded diameter minimum spanning tree problem. The bi-MDCST is a NP-hard problem and models several practical applications in transportation and network design. We propose a bi-objective multiflow formulation for the problem and effective multi-objective metaheuristics: a multi-objective evolutionary algorithm and a fast nondominated sorting genetic algorithm. Some guidelines on how to optimize the problem whenever a priority order can be established between the two objectives are provided. In addition, we present bi-MDCST polynomial cases and theoretical bounds on the search space. Results are reported for four representative test sets.

Keywords Spanning trees · Multiflow formulation · Multi-objective metaheuristics · Transportation and network design

1 Introduction

The bi-objective minimum diameter-cost spanning tree problem (bi-MDCST) is defined in a connected and undirected graph $G = (V, E)$ with a set V of vertices and a set E of edges. A cost $c_{ij} \geq 0$ is associated to each edge $[i, j] \in E$, with $i < j$. Let $\mathcal{T}$ be a spanning tree of G.

A. C. Santos (✉)
ICD-LOSI, Université de Technologie de Troyes, 12, rue Marie Curie, CS 42060,
10004 Troyes Cedex, France
e-mail: andrea.duhamel@utt.fr

D. R. Lima · D. J. Aloise
Universidade do Estado do Rio Grande do Norte, UERN, Rua Almino Afonso, 478,
CEP 59.610-210 Mossoró, RN, Brasil
e-mail: diegorocha@ufersa.edu.br

D. J. Aloise
e-mail: darioaloise@uern.br

Thus, there is a unique path $\mathcal{P}_{ij}$ in $\mathcal{T}$ linking any pair of nodes $i, j \in V$. Let d_{ij} be the number of edges in $\mathcal{P}_{ij}$. Then, the diameter D of $\mathcal{T}$ is defined as $D = max\{d_{ij} : i, j \in V, i \neq j\}$. A minimum spanning tree (MST) of G is a spanning tree $\mathcal{T}$ with minimum total cost. Finding an MST of a graph is a polynomial-time problem, as well as defining the diameter of a tree. This work focuses on the bi-MDCST that consists of finding a spanning tree of G with minimum total cost and minimum diameter.

The bi-MDCST is a NP-hard problem [14] and addresses network design and transportation logistic applications. In network design, the diameter refers to quality of service (QoS) requirements where small diameters reduce delays and improve reliability. Costs can represent time or financial costs to transmit data, to install the network infrastructure, etc. Another application appears on high speed trains where one looks for an MST backbone, and minimum diameters reduce the transportation time between any pairs of cities and improve QoS [32].

The bi-MDCST generalizes some problems such as the bounded diameter minimum spanning tree problem (BDMST) and the minimum diameter spanning tree problem (MDST). The BDMST consists of finding a spanning tree with minimum total cost, where the diameter does not exceed a given positive integer value. The BDMST is NP-hard [9] when $4 \leq D < |V|-1$, and several formulations [10,29], exact [11,22,23] and heuristics [19,24,27] methods are found in the literature. The MDST has received less attention, and seeks a spanning tree (not necessarily with minimum total cost) where the diameter is minimized. Distributed and efficient algorithms are presented in [2,13].

Some works in the literature deal with bi-objective spanning trees with the minimization of two cost functions such as the branch-and-bound framework proposed in [30], and the enumeration in two phases introduced in [26,31]. Moreover, the multi-objective evolutionary algorithm (MOEA) for the network design problem presented in [35] aims at minimizing the infrastructure cost as the first objective, and at minimizing the maintenance cost as the second one. The work [1] is also dedicated to optimizing two cost functions using a multi-objective greedy randomized adaptive search procedure. However, as far as we know, few works focus on the bi-MDCST with its two specific objective functions. A theoretical study is found in [14] and the authors prove several problems based on spanning trees are NP-hard, including the bi-MDCST. Approximation algorithms for a class of bicriteria network design problems are proposed in [20], but no computational experiments are reported for the bi-MDCST. A MOEA for the bi-MDCST is presented in [15,28]. Results are compared with constructive and greedy heuristics for the BDMST. However, such kinds of heuristics do not produce high-quality results for the BDMST. In this study, we propose formulations for the bi-MDCST using different strategies to deal with the two objectives. The first one considers a simultaneous optimization of the two objectives. For such a purpose, a general bi-objective multiflow formulation is proposed inspired by the work [10] for the BDMST, and two multi-objective metaheuristics: a MOEA and a fast nondominated sorting genetic algorithm (NSGAII). The MOEA is based on the one proposed in [28], with some improvements. The second strategy relies on optimizing the bi-MDCST whenever a priority order is established between the two objectives. Results are reported for four benchmarks of instances and several metrics have been used to evaluate the MOEA and the NSGAII algorithms.

This paper is organized as follows: A general multicommodity flow formulation is introduced in Sect. 2, followed by some simple polynomial solvable bi-MDCST cases in Sect. 3. Then, the next sections are dedicated to presenting strategies for solving the bi-MDCST. For instance, an optimization in two phases is presented in Sect. 4. A MOEA and a NSGAII metaheuristics are detailed in Sect. 5. Finally, computational results are reported in Sect. 6, and concluding remarks are given in Sect. 7.

2 A general multicommodity flow formulation for the bi-MDCST

The general multiflow formulation has been developed based on the work of Gouveia and Magnanti for the BDMST [10]. The formulation makes use of an undirected graph $G = (V, E)$ and the diameter $D = max\{d_{pq} : p, q \in V, p \neq q\}$, as mentioned before. Moreover, let x_{ij} be the decision variables on the choice of edge $[i, j]$. If edge $[i, j]$ belongs to the solution $x_{ij} = 1$, otherwise $x_{ij} = 0$. The flow variables are defined in the set A which is obtained from the set E by considering that for every edge $[i, j] \in E$ with $i < j$, there are arcs (i, j) and (j, i). Thus, the directed flow variables y_{ij}^{pq} specify if the path from p to $q, \forall p, q \in V, p \neq q, i \neq q$ and $j \neq p$, passes through arc $(i, j) \in A$, i.e. $y_{ij}^{pq} = 1$, otherwise $y_{ij}^{pq} = 0$. Obviously, $y_{ij}^{pi} = y_{ij}^{jq} = 0$ since no cycles exist in a tree. Let variables d_{pq} be the number of edges in the path from p to q.

The formulation from (1) to (12) formally defines the bi-MDCST. A solution is then a spanning tree $\mathcal{T}$ defined in a 2-dimensional space by its cost z_1 and its diameter z_2 values. Let $\mathcal{X}$ be the feasible space of solutions $\mathcal{T}$ defined by the vector $f(x) = \{z_1, z_2\}$. Considering the objectives $i = 1, 2$, a solution $f_i(x)$ dominates another $f_i(x')$ if and only if $f_i(x) \leq f_i(x')$ for all objectives $i = \{1, 2\}$ and $f_i(x) < f_i(x')$ for at least one of the objectives $i = \{1, 2\}$. A solution $\mathcal{T}_i \in \mathcal{X}$ is said to be Pareto-optimal $f_i(x^*)$ with respect to $\mathcal{X}$ if and only if there is no $\mathcal{T}_j \in \mathcal{X}$ for which $f_j(x)$ dominates $f_i(x^*)$. The Pareto-optimal front is the set of Pareto-optimal $f(x^*)$ solutions.

$$z_1 = \min \sum_{[i,j]\in E} c_{ij} \cdot x_{ij} \tag{1}$$

$$z_2 = \min \; D \quad \text{subject to} \tag{2}$$

$$\sum_{[i,j]\in E} x_{ij} = |V| - 1, \tag{3}$$

$$\sum_{j:(i,j)\in A} y_{ij}^{iq} = 1 \quad \forall i, q \in V, i \neq q, \tag{4}$$

$$\sum_{j:(i,j)\in A} y_{ij}^{pq} - \sum_{j:(j,i)\in A} y_{ji}^{pq} = 0 \quad \forall i, p, q \in V, p \neq q, i \neq q, i \neq p, \tag{5}$$

$$\sum_{i:(i,j)\in A} y_{ij}^{pj} = 1 \quad \forall p, j \in V, p \neq j, \tag{6}$$

$$y_{ij}^{pq} + y_{ji}^{pq} \leq x_{ij} \quad \forall [i, j] \in E, \forall p, q \in V, p \neq q, \tag{7}$$

$$\sum_{\substack{(i,j)\in A \\ i\neq q, j\neq p}} y_{ij}^{pq} \leq d_{pq} \quad \forall p, q \in V, p \neq q, \tag{8}$$

$$D \geq d_{pq} \quad \forall p, q \in V, p \neq q, \tag{9}$$

$$y_{ij}^{pq} \in \{0, 1\} \quad \forall (i, j) \in A, \forall p, q \in V, p \neq q, i \neq q, j \neq p, \tag{10}$$

$$x_{ij} \in \{0, 1\} \quad \forall [i, j] \in E, \tag{11}$$

$$d_{pq} \geq 1 \quad \forall p, q \in V, p \neq q. \tag{12}$$

The two objectives are gibven in Eqs. (1) and (2), and they aim respectively at minimizing the total cost and the diameter. Restriction (3) ensures the spanning tree has $|V| - 1$ edges. Restrictions (4), (5), and (6) are respectively responsible for sending, conserving, and receiving the flow. Inequalities (7) state no flow passes through edge $[i, j]$ whenever edge $[i, j]$

does not belong to the solution, i.e. $x_{ij} = 0$. Constraints (8) compute the number of edges in a path from p to q. Restrictions (8), (9) together with the objective function (2) minimize the diameter. Variables are defined from (9) to (12). This formulation contains $O(|A| \cdot |V|^2)$ variables and $O(|E| \cdot |V|^2)$ constraints.

In order to perform the diameter minimization, the following modifications have been made on the model proposed in [10]: the second objective function (2) has been introduced, the variables d_{pq} and constraints (9) have been added, and the right side of restrictions (8) are set to d_{pq} instead of D. It is important to highlight that variables d_{pq} can be dropped from the formulation. However, keeping them allows us to obtain the number of edges between each node p, q without any additional computation. This can be interesting for the development of methods coupling exact and heuristics strategies.

3 Some polynomial cases and theoretical bounds

Some Pareto-optimal solutions to the bi-MDCST can be computed in polynomial time following the results found by [9] for the BDMST problem. Consider the graph $G = (V, E)$ defined in Sect. 2. Moreover, the Property 1 defined by Handler [12] is applied in the proofs.

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.

Proposition 1 *Given the assumption that there are solutions for G with diameters $D = 2$ and $D = 3$ (a complete graph always has spanning trees of diameters $D = 2$ and $D = 3$), the Pareto-optimal spanning trees with diameters $D = 2$ and $D = 3$ are computed in polynomial time.*

Proof for the case when $D = 2$ According to Property 1 for the even D case, a spanning tree has a central node. Thus, a procedure can build, at each iteration, a spanning tree considering each node $c \in V$ at a time as the central node. All the other nodes $i \in V \backslash \{c\}$ are connected to c by adding edges $[i, c]$, if $i < c$, otherwise $[c, i]$. Building a unique spanning tree of $D = 2$ consumes $O(|V|)$. But at most $|V|$ spanning trees of diameter $D = 2$ are generated. Thus, the Pareto-optimal solution with $D = 2$ is computed with computational complexity $O(|V|^2)$. □

Proof for the case when $D = 3$ Given a minimum spanning tree $\mathcal{T}_j$ with $D = 2$. A spanning tree $\mathcal{T}_i$ with $D = 3$ is in the Pareto-optimal front if and only if the total cost of $\mathcal{T}_i$ is less than the cost of $\mathcal{T}_j$. Otherwise, $\mathcal{T}_i$ does not belong to the Pareto-optimal front since it is dominated by $\mathcal{T}_j$. Thus, for $D = 3$, the following hypothesis is taken into account: suppose the solution with $D = 3$ is not dominated. Then, finding such a Pareto-optimal solution can be done in polynomial computational time. Following Property 1 for the odd D case, a spanning tree has a central edge $[i, j] \in E$. Then, at each iteration, a procedure takes an edge $[i, j] \in E$ as the central one, and connects all other nodes $k \in V \backslash \{i, j\}$ to one of the central edge extremities i or j. The edge with minimum cost connecting k to one of the extremities i or j, is included in the spanning tree. Building a unique spanning tree with $D = 3$ consumes $O(|V|)$, but at most $|E|$ spanning trees are generated and compared in terms of total cost. Thus, the procedure has computational worst case complexity $O(|E| \cdot |V|)$ □

Concerning the MST of a given graph G, its cost is an upper bound on the search space considering the cost objective function. An MST is unique whenever all costs associated to each $[i, j] \in E$ are distinct, and it can be computed in polynomial time by using the classic algorithms of Prim and Kruskal [6]. On the contrary, whenever all (or some) costs are identical, in the context of the bi-MDCST, an enumeration of the minimum spanning trees is needed to chose the one with minimum diameter. A special bi-MDCST polynomial case occurs when all costs associated to each edge $[i, j] \in E$ are identical, and the graph G is complete. In this case, it is obvious that the Pareto-optimal front is degenerated and composed of exactly one solution defined by the cost of the MST, and with $D = 2$.

A lower bound considering the diameter objective function can be obtained from the graph G. The diameter of a graph differs from the diameter of a spanning tree since a spanning tree has a unique path between each pair of nodes. Denote by $d'_{ij}(i, j \in V)$ the minimum number of edges among of every possible paths between i and j in G, the diameter of G is $D' = max\{d'_{ij} : i, j \in V, i \neq j\}$ which is a lower bound on the search space considering the diameter. It is obvious that the minimum diameter spanning tree of G has a diameter greater than or equal to the diameter of G. The proposed multi-objective genetic algorithms use the cost of the MST and the diameter of G to bound the search space.

4 Solving the bi-MDCST

In practice, there are several ways to deal with the two objectives. The works [4,8,38] provide an overview of new trends, recent advances and bibliography reviews on multi-objective strategies. Some well-known strategies for solving multi-objective problems consist of aggregating the two objective functions, establishing a priority order to optimize the two objectives, or else optimizing simultaneously the two objectives.

In the aggregating strategy, after an appropriate normalization, the objective functions z'_1 and z'_2 can be aggregated to obtain $f(x)$. Thus, a linear combination $f(x) = \alpha \cdot z'_1 + (1-\alpha) \cdot z'_2$ can be applied. In this approach, the scalar $\alpha \in [0, 1]$ represents the compromise between the two objectives. Whenever $\alpha = 0$, only the diameter is considered, and if $\alpha = 1$ the total cost is optimized.

Optimizing the two objectives in two distinct phases implies the decision makers are able to determine a priority order for the objectives. It can be seen as a special case of the aggregating strategy where α is set to 0 or 1 in order to state the first objective to be optimized.

Finally, optimizing several objectives simultaneously can be achieved by applying methods such as metaheuristics, branch-and-bound, column generation, ϵ-constraint [34] and parallel partitioning method [18].

4.1 Optimization in two phases for the bi-MDCST

Using an optimization in two phases, the objectives are optimized in a priority order. Here, the diameter is used as the priority objective. Thus, it corresponds to the special case where $\alpha = 0$. In the second phase, the optimization relies on finding an MST where the diameter is bounded to the value found in the first optimization phase. Such a strategy is interesting for applications where small diameters are required. Moreover, one can use well-known formulations for the BDMST in the second optimization phase.

Formulation from (2) to (12) is used for the first optimization phase. As pointed out in [10,29], once the diameter is fixed, a single commodity network flow formulation can be applied for the BDMST. Then, in the second optimization phase, the formulations introduced in [29] are considered. In spite of the dual gaps, such formulations are able to prove optimality

for instances of medium size for the BDMST. These formulations make use of Property 1 previously defined in Sect. 3, and work on a digraph $G' = (V', A')$ that is obtained from $G = (V, E)$ as follows: an artificial vertex r is introduced in V. Thus, $V' = V \cup \{r\}$ and $A' = A \cup \{(r, 1), \ldots, (r, |V|)\}$, and for every edge $[i, j] \in E$, with $i < j$, arcs (i, j) and $(j, i) \in A'$ are added with costs $c_{ij} = c_{ji}$.

Both odd and even D formulations use the decision variables x_{ij} on the choice of arc $(i, j) \in A'$, and non-negative variables u_i which specify the number of arcs in a path from r to $i \in V'$. When D is odd, the formulation also uses binaries variables z_{ij} that define, whenever edge $[i, j] \in E$ is selected as the central spanning tree edge, $z_{ij} = 1$, or not $z_{ij} = 0$. Let D^* be the optimal diameter found in the first optimization phase. Thus, $L = D^*/2$ when D^* is even and $L = (D^* - 1)/2$ when D^* is odd. For D^* even, the second optimization phase is given as:

$$\min \sum_{(i,j)\in A} c_{ij} \cdot x_{ij} \quad \text{subject to} \tag{13}$$

$$\sum_{j\in V} x_{rj} = 1, \tag{14}$$

$$\sum_{(i,j)\in A'} x_{ij} = 1 \quad \forall j \in V, \tag{15}$$

$$u_i - u_j + (L+1)x_{ij} + (L-1)x_{ji} \leq L \quad \forall (i, j) \in A', \tag{16}$$

$$u_i \leq L + 1 \quad \forall i \in V', \tag{17}$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A', \tag{18}$$

$$u_i \geq 0 \quad \forall i \in V'. \tag{19}$$

Objective (13) minimizes the total cost. Restriction (14) states the artificial vertex r is connected to only one vertex in V. Constraints (15) ensure that only one arc must be incident to each vertex of V. Inequalities (16) define a topological order from the vertex r to each vertex $i \in V$ and eliminate subcycles. Restrictions (17) together with (16) establish that paths from the artificial vertex r to each vertex $i \in V$ have at most $L + 1$ arcs. Variables are defined in (18) and (19).

When D^* is odd, the formulation takes into account that the MST center is an edge as follows:

$$\min \sum_{(i,j)\in A} c_{ij} \cdot x_{ij} + \sum_{[i,j]\in E} c_{ij} \cdot z_{ij} \quad \text{subject to} \tag{20}$$

$$\sum_{j\in V} x_{rj} = 2, \tag{21}$$

$$\sum_{[i,j]\in E} z_{ij} = 1, \tag{22}$$

$$z_{ij} = x_{ri} \cdot x_{rj} \quad \forall [i, j] \in E, \tag{23}$$

$$z_{ij} \in \{0, 1\} \quad \forall [i, j] \in E, \tag{24}$$

Constraints (15)–(19).

The objective function (20) computes the total cost including the center edge cost. Restriction (21) ensures the artificial central vertex r is connected to exactly two vertices of V. Constraints (22) and (23) guarantee only one central edge is selected. Restrictions (23) are nonlinear, but they can be easily linearized as shown in [29]. Variables z_{ij} are defined in (24). Constraints (15)–(19) have already been defined. Readers are referred to [29] for further details.

5 Multi-objective metaheuristics

MOEA are genetic algorithms enclosing operators to classify non-dominated solutions. Several MOEA approaches are available in the literature [5] such as the strength pareto evolutionary algorithm (SPEA), the Niched–Pareto genetic algorithm (NPGA), the NSGAII, etc. They mainly differ in the operators applied to classify solutions as dominated or not, and on the genetic operators used to modify the population. MOEA strategies try to find (or to be close to) the Pareto-optimal front.

A pseudo-code of a standard MOEA algorithm is presented in Algorithm 1. It is the main procedure applied to the proposed MOEA and NSGAII. Let t and R_t be respectively the current iteration and the overall population at the iteration t. Moreover, the *ranking* corresponds to the number of solutions $\mathcal{T}_j$ which dominate a solution $\mathcal{T}_i$. The initialization steps from lines 1–5 mainly consist of generating an initial population R_1 (line 2) and evaluating each individual $i \in R_1$ taking into account the objective functions (line 3). Then, solutions are classified by using at least the *ranking* operator and positioning them in the search space as dominated or not (line 4). The Pareto front is built and kept in line 5. A number of evolutional iterations are performed from lines 6–14. Initially, elite (good) individuals are selected (line 7). Thus, genetic operators (mutation, crossover, recombination, etc) are applied using individuals belonging to the population in line 8, then t and the population are updated in lines 9 and 10, respectively. The new population is evaluated (line 11), solutions are classified (line 12), and the Pareto front is updated (line 13). The procedure in Algorithm 1 returns the best Pareto front found so far (line 15).

```
1  t ← 1;
2  Generate initial population R_t;
3  Evaluate the objectives values of each individual i ∈ R_t;
4  Classify each individual i ∈ R_t as dominated or not;
5  Update the best Pareto-front;
6  while (stopping criterion not met) do
7  |  Select a set of elite individuals from R_t;
8  |  Apply genetic operators using the elite individuals;
9  |  t ← t + 1;
10 |  Update the population R_t;
11 |  Evaluate the objective values of each individual i ∈ R_t;
12 |  Classify each individual i ∈ R_t as dominated or not;
13 |  Update the best Pareto-front;
14 end
15 return the best Pareto-front;
```

Algorithm 1: Generic MOEA pseudo-code.

5.1 Multi-objective genetic algorithms for the bi-MDCST

A standard MOEA and a NSGAII have been developed for the bi-MDCST. The MOEA [28] has been applied since, to our knowledge, it is the first bi-objective metaheuristic "dedicated" to the bi-MDCST. On the other hand, the NSGAII [7] has been shown to be a very

efficient heuristic for solving multi-objective problems [36]. The works [16,21] investigate the performance of genetic algorithms applied to multi-objective spanning tree problems.

The MOEA and the NSGAII proposed in this work share the following characteristics. (i) The population size is set to $N = 2 \cdot |V|$ solutions, and individuals are generated as described in the sequence. (ii) The data structure for a solution corresponds to a list of predecessors for each node in the tree. The works [3,25] provide details on data structure performance to encode trees. (iii) A 2-opt local search is performed to improve the solutions, and it is applied to every new individual in the incumbent population. A move in this local search consists of exchanging two edges in the solution by two edges not belonging to the solution. The move is performed in such a way that the diameter of the spanning tree does not change. Thus, a move is accepted whenever the cost reduces. The local search using the 2-opt neighborhood updates a given solution when the first improvement move is found, following a first-improvement search strategy.

As mentioned above, t and R_t are respectively the current iteration and the overall population at the iteration t. The population R_t is composed of sets P_t and Q_t. Thus, $R_t = P_t \cup Q_t$, where $|R_t| = N$ and $|P_t| = |Q_t| = N/2$. The set P_t of individuals is composed of $|V| - 2$ individuals randomly generated, and two particular solutions: an MST of G, and a spanning tree $\mathcal{T}$ of G with minimum diameter $D = 2$ or $D = 3$, if it exists. The MST is computed using Prim's algorithm [6]. The $|V| - 2$ solutions of P_t are built using a randomized Prim's algorithm as in [25]. Prim's algorithm uses a greedy strategy and the spanning tree is built by adding one smallest edge cost at a time. In the Prim randomized version, one edge is randomly chosen to enter the solution among every edge with an extremity node in the incumbent spanning tree, and another node outside of the solution. The set Q_t is obtained using genetic operators between two solutions randomly chosen in P_t.

The MOEA for the bi-MDCST proposed in [28] identifies non-dominated solutions by using the *ranking*, while the NSGAII applies the *ranking*, and the *crowding distance* that is a distance of solutions around a specific solution.

In terms of stopping criteria, calibration experiments have been performed to fine-tune it. Other specific details for the MOEA and the NSGAII are given below.

5.1.1 A standard MOEA for the bi-MDCST

The MOEA suggested in [28] for the bi-MDCST uses an edge-set data structure (a list of edges belonging to the spanning tree) and works as follows: the initial population has $|V|$ individuals. Then, individuals in the population are evaluated and classified. At each iteration, only two new individuals are introduced in the population. They are generated by applying *recombination* and a *modified edge-delete mutation genetic* operators, detailed below. After that, the population is ordered and the two worst individuals are discarded. Convergence is controlled by the ranking histogram suggested in [17]. The algorithm stops when the rank histogram reaches a pre-specified target rate.

In the *recombination* operator, a new solution $\mathcal{T}_1$ is derived from two others, $\mathcal{T}_2$ and $\mathcal{T}_3$, by taking an edge at a time belonging to both $\mathcal{T}_2$ and $\mathcal{T}_3$, and including them in $\mathcal{T}_1$. If necessary, edges randomly chosen from $\mathcal{T}_2$ or $\mathcal{T}_3$ are added to $\mathcal{T}_1$, without making cycles. In the *greedy edge replacement mutation* operator, an edge $[i, j] \in E$ is randomly deleted in the spanning tree $\mathcal{T}$. Thus, $\mathcal{T}$ is disconnected, generating two subtrees t_1 and t_2. These subtrees are connected with the smallest cost edge $[k, l] \in E$ linking t_1 and t_2, where $[k, l] \neq [i, j]$. The *modified edge delete mutation* operator is a combination of the *greedy edge replacement mutation* and the *recombination* operators. Initially, an edge is removed from $\mathcal{T}_1$ which makes it unfeasible. Thus, a recombination is performed using $\mathcal{T}_1$ and a feasible solution $\mathcal{T}_2$. In a

previous work [15], the authors mention that the individuals are randomly selected in the population to generate new ones.

Some details of the MOEA presented in [28] are omitted, for example, how the operators have been applied along the MOEA iterations. Thus, it is difficult to exactly reproduce such an algorithm. Moreover, some aspects of the MOEA [15,28] make the approach non competitive. For instance, only two new solutions enter the population at each iteration. As a consequence, the convergence and the population diversity are significantly affected. In order to try to overcome this problem, we use a similar strategy as applied for the NSGAII. Thus, here, the population is improved first by setting a size of $N = 2 \cdot |V|$, and second, by replacing up to $|V|$ individuals at each iteration. The *ranking* operator and the *genetic operators* have been considered as in the work [28]. Furthermore, we use a list of predecessors to encode the spanning trees, as explained in Sect. 5.1.

In the first iteration of the MOEA, the population $R_t = P_t \cup Q_t$ is obtained by randomly generating solutions for the set P_t, and by applying the *modified edge delete mutation* operator to build the set Q_t. For the other iterations, $|V|$ elite solutions are selected and the set Q_t of individuals is obtained as in the first iteration.

5.1.2 A NSGAII for the bi-MDCST

The NSGAII for the bi-MDCST is inspired by the steady-state algorithm proposed in [7]. Such an algorithm uses two operators to classify solutions, the *ranking* and the *crowding distance*. For each individual belonging to the population, the *ranking* computes the number of solutions which dominate it. Thus, the individuals with *ranking* equal to zero are set in the best Pareto front found so far. After applying the *ranking*, all individuals in the population are set in a Pareto front on the search space. For each individual, the *crowding distance* defines the distance to its nearest neighbours in the same Pareto front and it contributes to making the Pareto front uniform. Thus, the bigger the *crowding distance* is, the more solutions should be generated in the region considered.

In the first iteration of the NSGAII, the population $R_t = P_t \cup Q_t$ is obtained by randomly generating solutions for the set P_t, and by applying the crossover operator for trees, proposed in [3], to build the set Q_t. The crossover works as follows: two solutions, one from the elite set $\mathcal{T}_j$ and another from the rest of the population $\mathcal{T}_i$ are randomly selected. Then, a support graph $G'' = (V, E'')$ is built, where every edge $[i, j] \in \mathcal{T}_i$ and every edge $[k, l] \in \mathcal{T}_j$ are added to E''. Figure 1 illustrates an example, where solutions $\mathcal{T}_i$ and $\mathcal{T}_j$ are respectively depicted in Fig. 1a, b, and the resulting support graph G'' is given in Fig. 1c. After building G'', a spanning tree of G'' is computed using the randomized Prim algorithm. This operator allows us to avoid unfeasible trees, particularly for sparse graphs. For the other iterations, the $|V|$ better solutions pass through the next iteration, and the set Q_t is generated as mentioned above.

Once the initial population R_t is built, the *ranking* and the *crowding distance* operators are computed for every solution $\mathcal{T}_i \in R_t$. Let the *crowding distance* w be a vector of size

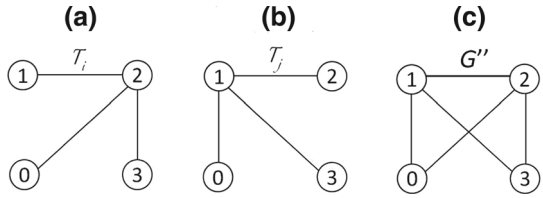

Fig. 1 Example of a support graph G''

M. It is computed as shown in Eq. (25), where $suc(j)$ and $pred(j)$ are respectively the value which precedes j in w and the value which follows j in w. The bi-MDCST involves two objectives $m = 1, 2$, thus z_m^{max} and z_m^{min} are the maximum and the minimum objective functions values. Since cost and diameter are values with different scales, values have been properly normalized in the interval [0, 1].

$$w_j = \sum_{m=1}^{M} \left(\frac{z_m^{suc(j)} - z_m^{pred(j)}}{z_m^{max} - z_m^{min}} \right) \tag{25}$$

Solutions in the population are then ordered using the *ranking* and the *crowding distance*. Given $\mathcal{T}_i$ and $\mathcal{T}_j$, its corresponding *ranking* r_1 and r_2, and its *crowding distance* w_i and w_j. A solution $\mathcal{T}_i$ is better than $\mathcal{T}_j$ if the following conditions $(r_i < r_j)$ or $((r_i = r_j)$ and $(w_i > w_j))$ hold. After ordering solutions, the first $|V|$ best are kept to pass though the next iteration. The other $|V|$ individuals are generated using the crossover mentioned above. This procedure is performed until the stopping criterion is met. The best Pareto front found so far is returned.

6 Computational experiments

The experiments were performed on an Intel Core i7 with 2.7 GHz clock and 8Gb of RAM memory. Experiments for the optimization in two phases were performed using CPLEX 12 under default parameters. The standard MOEA and the NSGAII were developed in C ANSI with DevCpp 4.9.9.2. Results for the optimization in two phases are reported in Sect. 6.1. Then, Sect. 6.2 is dedicated to describing the results produced by the metaheuristics. Initially, a set of experiments has been addressed to calibrate both metaheuristics MOEA and NSGAII in Sect. 6.2.1. Then, extensive experiments are done to compare both metaheuristics 6.2.2. Finally, Sect. 6.2.3 shows a comparison of the metaheuristics with the optimal results produced by using the optimization in two phases.

Four sets of instances are used in the experiments, three G, L and R are respectively from the published papers [10,19,24]. It is important to mention that such sets of instances are difficult and representative for both the BDMST and for the bi-MDCST. Considering the BDMST for which the search space is usually smaller than the one for the bi-MDCST, from our knowledge, no algorithm has proved optimality for instances proposed by [24]. Moreover, the instances of [10] have a number of edges with similar costs which raises the combinatorial choices of such problems.

The last set, referred here as T, has been developed in this work and it contains sparse instances composed of two subsets c and p of 9 and 11 instances respectively. The first 9 instances have been generated using an arbitrary *Hamiltonian cycle* to ensure the graph connectivity. The remaining edges are randomly added accordingly to the graph density set to {0.2, 0.3, 0.4}. For the other 11 instances, the graph connectivity is ensured by an arbitrary *Hamiltonian path*. The remaining edges are randomly added and the graphs have density {0.08, 0.09, 0.1}. In the naming format X_vY_Z, X corresponds to the instance type (c or p), Y to the number of vertices, and Z to the graph density.

6.1 Results for the optimization in two phases

Table 1 gives the results for the optimization in two phases, presented in Sect. 4. Each line corresponds to an instance, and the instance names, the MST diameter (D) and cost (C) are

Table 1 Results for the optimization in two phases

Instances	MST		First optimization phase				Second optimization phase			
	D	C	D*	LR	Time	Nodes	C*	LR	Time	Nodes
c_v15_0.2	11	433	7	4.1	76.9	2,375	497	398.3	0.2	171
c_v15_0.3	10	415	4	4.0	74.6	83	635	392.3	1.4	10,659
c_v15_0.4	10	375	4	3.0	30,862.2	20,250	465	341.0	0.8	2,972
c_v20_0.2	10	776	5	4.0	29,123.0	14,215	980	748.2	1.0	3,072
c_v20_0.3	8	470	5	3.1	326,655.7	72,378	618	436.7	1.7	4,412
c_v20_0.4	10	467	–	3.0	–	–	–	–	–	–
c_v25_0.2	12	599	–	4.0	–	–	–	–	–	–
c_v25_0.3	9	529	–	3.0	–	–	–	–	–	–
c_v25_0.4	12	447	–	3.0	–	–	–	–	–	–
p_v25_0.09	18	1,144	17	11.0	208	2,885	1,146	1,062.0	0.1	47
p_v25_0.10	16	1,186	9	8.0	41	277	1,323	1,099.0	0.2	783
p_v30_0.08	20	1,838	12	9.0	312.5	920	1,858	1,822.3	0.2	550
p_v30_0.09	14	1,217	9	7.0	49,340.0	32,321	1,407	1,161.1	2.1	10,127
p_v30_0.10	17	1,140	9	7.0	271,062.5	43,341	1,315	1,088.0	3.9	23,736
p_v35_0.08	20	1,604	–	8.0	–	–	–	–	–	–
p_v35_0.09	10	1,513	–	6.0	–	–	–	–	–	–
p_v35_0.10	14	1,112	–	7.0	–	–	–	–	–	–
p_v40_0.08	15	1,553	–	8.0	–	–	–	–	–	–
p_v40_0.09	13	1,467	–	5.2	–	–	–	–	–	–
p_v40_0.10	19	1,313	–	5.0	–	–	–	–	–	–

shown. Then, results for the first and the second optimization phases are depicted. The optimal diameter (D*) and the optimal cost (C*) are respectively provided, whenever possible, for the first and the second optimization phases. Moreover, the linear relaxation (LR), the running time (time) in seconds to prove optimality, and the number of nodes (*nodes*) visited in the branch-and-bound tree to prove optimality are given for both phases. The symbol (−) means the solver runs out of memory. Thus, some instances are not solved to optimality even for the first optimization phase. For such instances, the linear relaxation in the second optimization phase has not been computed since D^* values are unknown.

Results for the first optimization phase indicate the general multiflow formulation is able to treat only small and sparse instances. The running time increases significantly for small instances in this test set. Moreover, results show expected behavior: the tree diameter reduces when the graph density increases, and the problem becomes harder to solve. Formulations introduced in Sect. 4 for the second optimization phase solve the problem in a small running time. Thus, further research should be addressed in order to adapt such formulations for dealing with the first optimization phase.

6.2 Results for the multi-objective metaheuristics

Experiments for the multi-objective metaheuristics have been performed to calibrating and comparing them, as well as evaluating the quality of the solutions. Initially, a calibration is done to decide the stopping criteria. Then, results produced by the two metaheuristics MOEA

and NSGAII are compared. The algorithms' convergence is also analysed. Finally, optimal values from works [11,29] and from the multiflow formulation have been used to measure their performance.

Three metrics are used to evaluate the results. The first one Q indicates the number of solutions in the Pareto front. The second metric S, called of spacing, is a measure of spread (distribution) of solutions in the Pareto front found so far, and has been computed as in [33]. An average Euclidean distance between solutions is computed to get S. The smaller the value of S, the better the solutions distribution is because it indicates a uniform distribution in the best Pareto front. The third metric is the hypervolume H [37] which gives the area size from the worst solution θ in the search space. For the bi-MDCST, the solution θ corresponds to the diameter of an MST of G, and the cost of an MDST of G. The bigger the H value, the better the Pareto front is because it shows the front is far from θ. It is important to highlight that the values computed for all metrics have been normalized in the interval [0, 1].

6.2.1 Calibration experiments for the metaheuristics

Calibration experiments are presented in Tables 2 and 3 for a sample of instances coming from the four test sets. Instances se_v40_a400 and se_v60_a600 are from [10], c_v20_a190 and c_v25_a300 are from [19], $c\_v70\_d7\_1$ and $c\_v100\_d10\_1$ comes from [24], and $c\_v35\_0.08$ and $c\_v40\_0.10$ are proposed in this work. The other instances have similar results as for this sample. In order to observe the quality of solution, convergence, and running time, a number of 100, 300, and 500 independent iterations have been performed using the MOEA and the NSGAII. For each independent run, the values of metrics Q, S, H, and the running time (*time*) in seconds are given.

For both MOEA and NSGAII, in most of the cases, the values of Q improve when the number of iterations increases. However, for some instances, such as c_v20_a190 using the MOEA strategy, the Q value decreases when performing 500 iterations instead of 300. This happens when a better solution in the first front is found and it dominates the others. In this case, the H value increases which confirms such a result.

The expected behavior is that H values increase whenever more iterations are performed. It means the best Pareto front found so far is improved and solutions become far from θ.

Table 2 Calibration results for the MOEA

MOEA												
Instances	100 iterations				300 iterations				500 iterations			
	Q	S	H	Time	Q	S	H	Time	Q	S	H	Time
se_v40_a400	7	0.25	0.56	0.76	10	0.24	0.64	2.25	8	0.29	0.65	3.19
se_v60_a600	12	0.21	0.61	2.64	11	0.26	0.69	7.34	11	0.26	0.69	1.22
c_v20_a190	6	0.20	0.55	0.09	9	0.12	0.61	0.30	8	0.12	0.64	0.46
c_v25_a300	4	0.45	0.36	0.16	5	0.37	0.46	0.41	5	0.36	0.48	0.61
c_v70_d7_1	8	0.28	0.38	4.26	10	0.25	0.50	12.39	10	0.20	0.57	27.10
c_v100_d10_1	6	0.48	0.28	14.62	8	0.41	0.36	39.78	8	0.40	0.39	68.01
c_v35_0.08	6	0.18	0.54	0.68	8	0.09	0.55	2.08	8	0.09	0.58	3.30
c_v40_0.10	5	0.38	0.35	1.16	5	0.41	0.37	3.51	4	0.48	0.42	5.37

Table 3 Calibration Results for the NSGAII

NSGAII												
Instances	100 iterations				300 iterations				500 iterations			
	Q	S	H	Time	Q	S	H	Time	Q	S	H	Time
se_v40_a400	14	0.11	0.81	1.08	14	0.11	0.82	3.96	14	0.11	0.82	6.74
se_v60_a600	17	0.10	0.85	3.95	20	0.10	0.86	14.05	21	0.10	0.86	24.19
c_v20_a190	9	0.25	0.77	0.15	10	0.24	0.78	0.45	11	0.16	0.79	0.78
c_v25_a300	12	0.17	0.74	0.24	12	0.17	0.75	0.76	12	0.17	0.75	1.32
c_v70_d7_1	16	0.13	0.80	4.75	18	0.12	0.83	17.80	19	0.13	0.84	31.97
c_v100_d10_1	14	0.10	0.78	12.80	28	0.08	0.86	52.03	30	0.08	0.87	99.64
c_v35_0.08	8	0.11	0.66	1.05	7	0.12	0.69	3.47	7	0.12	0.69	5.79
c_v40_0.10	9	0.13	0.77	1.41	10	0.13	0.78	4.59	10	0.14	0.78	7.64

In many cases, the H values improve when passing from 300 to 500, or else it remains the same.

The spacing S values reduce when solutions are uniformly distributed in the Pareto front. In most of the cases, results obtained by applying the NSGAII behave like that. However, for the MOEA, the S values often grow when the number of iterations increases. It means the Pareto front is not uniformly filled. A similar result is obtained for a few particular instances like $c\_v40\_0.10$ using both strategies. In this case, the instance is very sparse and does not have many different solutions in the Pareto front. This conclusion comes from the fact that diameters are integer values. Then, for a small and sparse instances, the number of different solutions in the Pareto front can be limited.

Figures 2 and 3 present the initial population, and the best Pareto front after 100, 300, and 500 independent runs, respectively, for the MOEA and the NSGAII. Similar results are obtained for the other instances. $|V| - 2$ solutions in the initial population are randomly generated and genetic operators are applied to get the other $|V|$ solutions, which explains such a distribution in the search space for the first iteration. The best Pareto front found with 100 iterations is sometimes significantly worse compared to the runs with 300 and 500 iterations. This is the case for the instance se_v60_a600 with the MOEA, and for the instance $c\_v100\_d10\_1$ using the NSGAII. Passing from 300 to 500 slightly improves the Pareto front, but it allows us to make it uniform such as for instance se_v60_a600 solved by the NSGAII.

The MOEA and the NSGAII have been developed with similar population size and data structure to encode individuals (trees). However, it is expected that the running time to perform the same number of iterations differs since they have different operators to classify the population and different genetic operators. The objective remains to give the same opportunity in terms of running time for both strategies while performing at least 500 iterations. Then, for the comparison experiments between MOEA and NSGAII, the stopping criteria have been set as the maximum running time (rounded up) to compute 500 iterations using one of the strategies, considering the number of vertices of the instances for each test set.

6.2.2 Comparison experiments between the MOEA and the NSGAII

Computational results comparing the MOEA and the NSGAII are presented in Tables 4, 5, 6 and 7, respectively for the sets G, L, R, and T. Each line corresponds to an instance, and

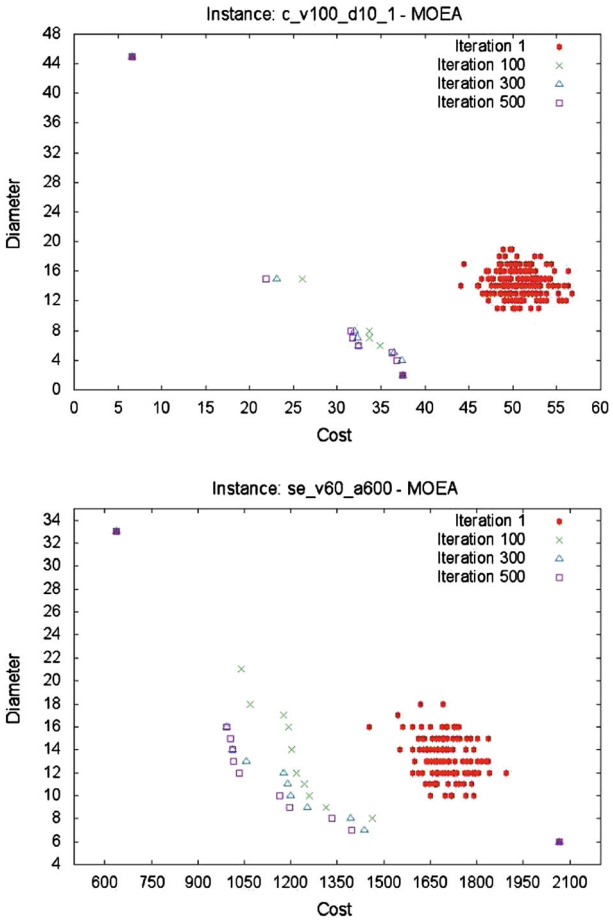

Fig. 2 Population evolution for the MOEA strategy

the two first columns depict its name and the running time in seconds. Then, metrics Q, S, and H values are shown for both MOEA and NSGAII. Symbol $(-)$ means the metric values have not been computed since there are less than or equal to two solutions in the Pareto front. Values in bold denote whenever the metric H is better for one of the strategies. We highlight values for H since it is a pertinent metric of the Pareto front quality. The higher the H values are, the further from θ the Pareto front is.

Considering the number of solutions Q in the Pareto front, for the sets G and R, respectively in Tables 4 and 6, the NSGAII outperforms the MOEA for all instances. For the test set L (Table 5), the NSGAII finds better results for 17 out of 24 instances, and has obtained similar results as the MOEA for 7 instances. The results for test set T in Table 7 slightly differ from the previous ones. The NSGAII finds better results than the MOEA for 8 out of 20 instances, and 9 similar results. The MOEA produces three better results than the NSGAII.

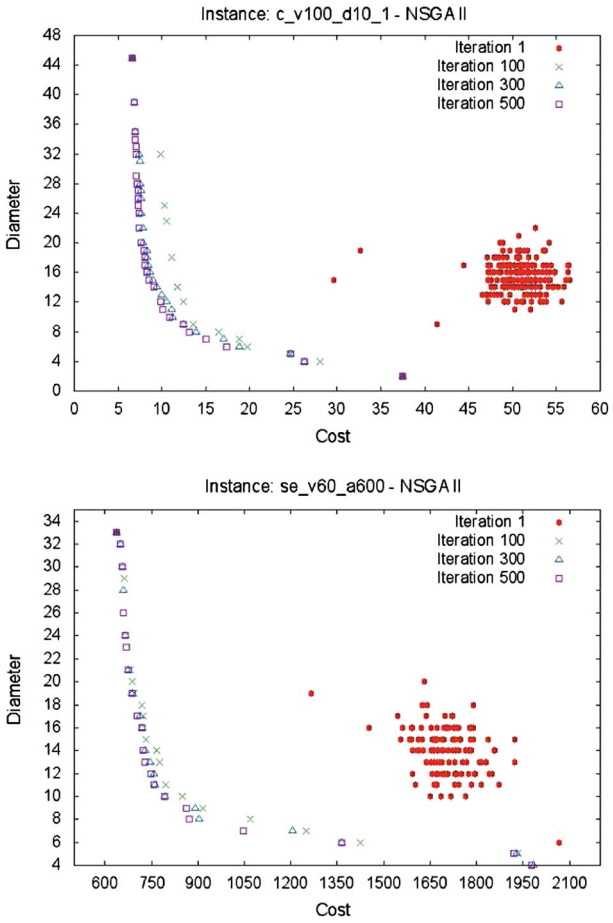

Fig. 3 Population evolution for the NSGAII strategy

Table 4 Results for the MOEA and the NSGAII using instances from [10]

Instances	Time	MOEA			NSGAII		
		Q	*S*	*H*	*Q*	*S*	*H*
se_v40_a400	60	10	0.18	**0.82**	13	0.13	0.81
sr_v40_a400	60	7	0.14	0.61	8	0.16	**0.65**
se_v60_a600	100	10	0.27	0.76	19	0.15	**0.88**
sr_v60_a600	100	6	0.17	0.48	13	0.20	**0.79**

The NSGAII has got better *S* values than the MOEA for 2 out of 4, 16 out of 24, 19 out of 20, and 12 out of 20, respectively for results in Tables 4, 5, 6, and 7. It is important to mention that the position of a solution in the search space will significantly impact on the spacing *S* values. Sometimes, even if *Q* and *H* values improve, the spacing can be worse. This is the case of the instance *s_v*20_*a*50_*d*7 (Table 5). It shows that solutions are not uniformly distributed in the Pareto front.

Table 5 Results for the MOEA and the NSGAII using instances from [19]

Instances	Time	MOEA			NSGAII		
		Q	*S*	*H*	*Q*	*S*	*H*
c_v10_a45_d4	10	7	0.14	0.60	7	0.12	**0.62**
c_v10_a45_d5	10	3	0.68	0.23	4	0.29	**0.54**
c_v10_a45_d6	10	4	0.51	0.46	4	0.28	**0.55**
c_v10_a45_d7	10	7	0.14	0.61	7	0.14	**0.64**
c_v10_a45_d8	10	5	0.17	0.56	5	0.17	0.56
c_v10_a45_d10	10	6	0.14	0.63	6	0.24	**0.66**
c_v15_a105_d4	15	8	0.25	0.72	10	0.18	**0.78**
c_v15_a105_d5	15	8	0.25	0.72	10	0.18	**0.78**
c_v15_a105_d8	15	5	0.27	0.55	6	0.24	**0.65**
c_v15_a105_d10	15	8	0.25	0.72	10	0.18	**0.78**
c_v25_a300_d4	25	12	0.16	0.72	12	0.17	**0.79**
c_v25_a300_d5	25	11	0.21	0.70	12	0.19	**0.81**
c_v25_a300_d6	25	12	0.18	0.72	13	0.17	**0.81**
c_v25_a300_d8	25	11	0.14	0.73	13	0.17	**0.84**
c_v25_a300_d9	25	8	0.24	0.61	9	0.19	**0.73**
s_v20_a50_d4	20	7	0.20	0.74	8	0.18	**0.75**
s_v20_a50_d5	20	7	0.34	0.66	8	0.24	**0.75**
s_v20_a50_d6	20	7	0.15	0.60	7	0.16	**0.65**
s_v20_a50_d7	20	6	0.16	0.61	7	0.29	**0.77**
s_v20_a50_d8	20	6	0.16	0.51	7	0.20	**0.66**
s_v40_a100_d4	30	12	0.19	0.84	17	0.16	**0.85**
s_v40_a100_d5	30	8	0.33	0.71	12	0.10	**0.79**
s_v40_a100_d6	30	8	0.23	0.64	9	0.15	**0.76**
s_v60_a150_d5	40	7	0.29	0.50	13	0.14	**0.73**

In terms of H values, the NSGAII produces better results than the MOEA for 61 out of 68 instances for the four test sets. It is worse only for three instances. The NSGAII found Pareto fronts of good quality for most of the cases. The *crowding distance* operator plays an important role since it manages the population diversity very well. As a consequence, it helps to improve the Pareto front along the NSGAII iterations.

Figure 4 illustrates the best Pareto front obtained by using the proposed MOEA and the NSGAII, and optimal values for some diameters for the BDMST from [11,29]. We notice that the NSGAII manages to find some optimal values for the BDMST or else to be close to them. It is important to highlight that the search space for bi-MDCST is larger than the one for the BDMST problems. Thus, it is an interesting and promising result, especially because the methods proposed in [11,29] are very sophisticated exact algorithms for the BDMST. The NSGAII produces good quality results for the bi-MDCST, and it works consistently well.

Figure 5 illustrates the best Pareto front for the NSGAII and the MOEA using a sample of instances from different test sets. It clearly shows the MOEA is not able to find solutions whenever the diameter increases, as it is the case for instances se_v40_a400 and $c\_v70\_d7\_3$. An opposite problem happens when the MOEA is not able to find solutions for small diameters

Table 6 Results for the MOEA and the NSGAII using instances from [24]

Instances	Time	MOEA			NSGAII		
		Q	*S*	*H*	*Q*	*S*	*H*
c_v50_d5_1	60	14	0.16	**0.77**	22	0.06	0.74
c_v50_d5_2	60	18	0.11	0.72	22	0.13	**0.84**
c_v50_d5_3	60	12	0.15	0.64	18	0.11	**0.80**
c_v50_d5_4	60	11	0.14	0.57	16	0.12	**0.70**
c_v50_d5_5	60	12	0.13	0.63	14	0.08	**0.74**
c_v70_d7_1	120	12	0.14	0.63	18	0.10	**0.81**
c_v70_d7_2	120	13	0.20	0.76	29	0.13	**0.89**
c_v70_d7_3	120	15	0.17	0.76	24	0.09	**0.85**
c_v70_d7_4	120	9	0.23	0.60	23	0.11	**0.81**
c_v70_d7_5	120	14	0.11	0.69	21	0.10	**0.81**
c_v100_d10_1	180	8	0.30	0.31	30	0.10	**0.88**
c_v100_d10_2	180	10	0.29	0.64	29	0.09	**0.88**
c_v100_d10_3	180	9	0.31	0.62	18	0.14	**0.84**
c_v100_d10_4	180	8	0.28	0.49	26	0.10	**0.84**
c_v100_d10_5	180	10	0.33	0.71	25	0.10	**0.86**
c_v250_d15_1	450	2	–	–	17	0.21	**0.76**
c_v250_d15_2	450	2	–	–	21	0.16	**0.84**
c_v250_d15_3	450	3	0.32	0.22	14	0.26	**0.72**
c_v250_d15_4	450	5	0.50	0.32	19	0.18	**0.78**
c_v250_d15_5	450	6	0.40	0.30	16	0.17	**0.81**

as illustrated by the instance *c_v*50_*d*5_1. In any case, we notice the main problem of the MOEA strategy is managing the population diversity.

6.2.3 Comparison experiments between the metaheuristics and the optimization in two phases

Results produced by the optimization in two phases are compared to the corresponding point in the Pareto front produced by the MOEA and by the NSGAII. Table 8 shows, for each instance, the optimal solution values defined by its diameter D^* and its cost C^*, followed by the total running time in seconds to perform the two optimization phases. It important to mention that such running time is the one required to compute a single solution. The Pareto front is composed of several solutions. Considering the diameter D^*, a deviation in percentage between the optimal cost value and the best cost value found so far by the MOEA and by the NSGAII for the corresponding diameter is depicted in the columns named gap. The symbol "*" means the metaheuristic did not return a result for the corresponding diameter. This situation can occur since the heuristics discard dominated solutions considering the final population. The last column *time* depicts the running time in seconds given for both MOEA and NSGAII.

The NSGAII manages to find optimal values for the corresponding diameter for 8 out of 10 instances. Clearly, the NSGAII and the MOEA fail to obtain solutions when $D = 5$ which

Table 7 Results for the MOEA and the NSGAII using instances proposed here

Instances	Time	MOEA			NSGAII		
		Q	S	H	Q	S	H
c_v15_0.2	10	2	–	–	3	0.29	**0.64**
c_v15_0.3	10	5	0.07	0.59	5	0.09	**0.60**
c_v15_0.4	10	5	0.11	0.57	5	0.23	**0.59**
c_v20_0.2	15	4	0.25	0.29	4	0.13	**0.41**
c_v20_0.3	15	2	–	–	3	0.47	**0.48**
c_v20_0.4	15	5	0.11	0.49	6	0.17	**0.61**
c_v25_0.2	20	6	0.34	0.58	6	0.14	**0.59**
c_v25_0.3	20	3	0.52	0.29	4	0.31	**0.70**
c_v25_0.4	20	7	0.31	0.69	7	0.19	0.69
p_v25_0.08	20	2	–	–	2	–	–
p_v25_0.09	20	6	0.15	0.66	6	0.14	0.66
p_v30_0.08	25	4	0.29	0.49	4	0.13	**0.59**
p_v30_0.09	25	4	0.06	0.30	5	0.13	**0.44**
p_v30_0.10	25	6	0.23	0.52	7	0.11	**0.66**
p_v35_0.08	30	8	0.21	0.65	6	0.16	**0.77**
p_v35_0.09	30	2	–	–	1	–	–
p_v35_0.10	30	5	0.23	0.62	6	0.29	**0.71**
p_v40_0.08	35	5	0.21	0.36	4	0.28	**0.58**
p_v40_0.09	35	4	0.51	0.45	4	0.27	**0.55**
p_v40_0.10	35	8	0.20	0.29	9	0.16	**0.77**

is a not suprising result since small diameters $D = 4$ and $D = 5$ are known to be difficult to solved.

In spite of the gap obtained for the instance $c\_v20\_0.3$, the NSGAII found optimal values for all other instances in a much smaller running time compared to the multiflow formulation.

7 Concluding remarks and perspectives

In this work, a general multiflow formulation is proposed for the bi-MDCST inspired by the BDMST formulation proposed in [10]. To the best of our knowledge, it is the first mathematical formulation using multiflow for the bi-MDCST. Some polynomial solvable cases and bounds on the search space are presented. Moreover, this work also brings important contributions in terms of metaheuristics, a MOEA and a NSGAII metaheuristics have been developed for the bi-MDCST.

In addition, an optimization in two phases has been introduced, considering the diameter as the priority objective. Optimal values and lower bounds have been computed for sparse instances. The results indicate the general multiflow formulation is able to solve sparse and small instances. Moreover, they show expected behavior: when density increases, the tree diameter reduces and the problem is harder to solve.

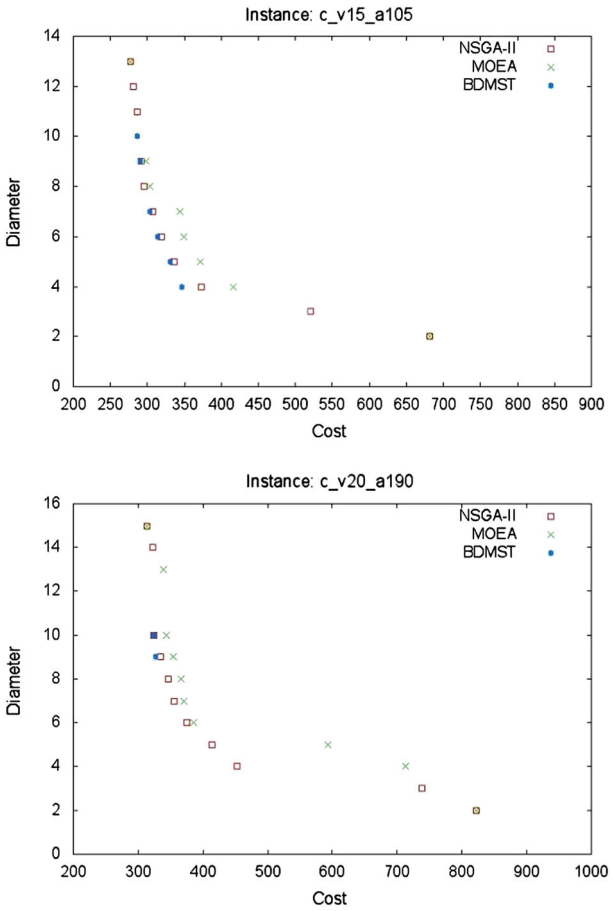

Fig. 4 Comparison experiments between the MOEA, the NSGAII, and optimal solutions for the BDMST

In terms of metaheuristics, the proposed MOEA is inspired by the work [28], with improvements in the size of population and data structures to encode trees. However, results indicate such a strategy fails to manage the population diversity. The NSGAII proposed in [7] has been applied to the bi-MDCST. The results show the NSGAII produces very good quality results for the bi-MDCST, and it works consistently well compared to the MOEA. Moreover, we also point out that the results produced by the NSGAII are very close to those produced by sophisticated exact methods for the BDMST, for some instances. This is a very interesting and promising result. The better performance of the NSGAII is probably due to the two operators used for evaluating and selecting individuals in the population. They allow an accurate management of diversity.

Regarding future work, there is room for improvements of the proposed bi-MDCST mathematical formulation, as well as investigating other formulations. Exact algorithms to solve the bi-MDCST such as branch-and-bound, and the Parallel Partitioning Method can also be developed. Moreover, we intend to improve the NSGAII by adding new genetic operators and new moves to the local search procedure. Other multi-objective metaheuristics could also be developed.

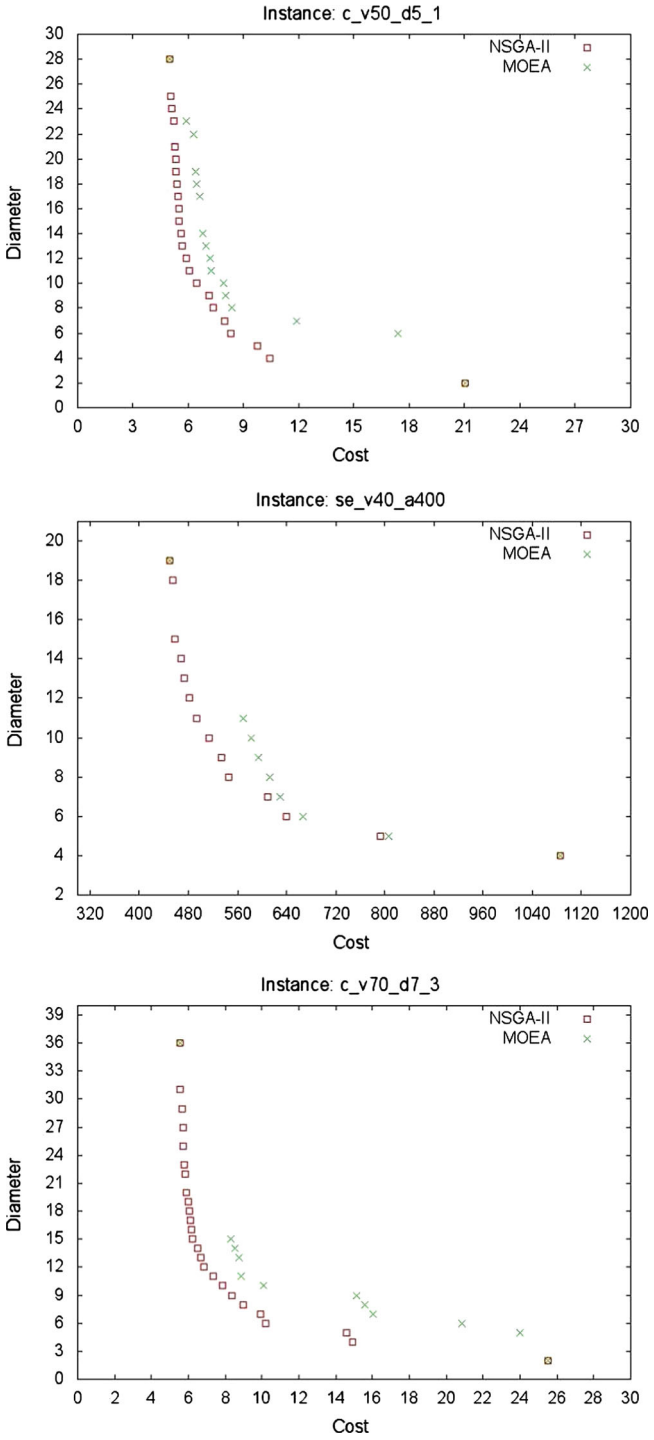

Fig. 5 Comparison experiments between the MOEA and the NSGAII

Table 8 Deviation of the MOEA and the NSGAII compared to the optimization in two phases

Instances	Optimization in phases			MOEA	NSGAII	Time
	D*	C*	Time	Gap	Gap	
c_v15_0.2	7	497	77.1	0.00	0.00	10
c_v15_0.3	4	635	76.0	0.00	0.00	
c_v15_0.4	4	465	30,863.00	10.06	0.00	
c_v20_0.2	5	980	29,124.0	*	*	15
c_v20_0.3	5	618	326,657.4	*	4.63	
p_v25_0.09	18	1,146	208.1	0.00	0.00	20
p_v25_0.10	16	1,323	41.2	0.00	0.00	
p_v30_0.08	20	1,858	312.7	0.00	0.00	25
p_v30_0.09	14	1,407	49,342.1	*	0.00	
p_v30_0.10	17	1,315	272,150.5	7.85	0.00	

References

1. Arroyo, J.E.C., Vieira, P.S., Vianna, D.S.: A GRASP algorithm for the multi-criteria minimum spanning tree problem. Ann. Oper. Res. **159**, 125–133 (2008)
2. Bui, M., Butelle, F., Lavault, C.: A distributed algorithm for constructing the minimum spanning tree problem. J. Parallel Distrib. Comput. Arch. **64**, 571–577 (2004)
3. Carrano, E.G., Fonseca, C.M., Takahashi, R.H.C., Pimenta, L.C.A., Neto, O.M.: A preliminary comparison of tree encoding schemes for evolutionary algorithms, pp. 1969–1974. Montreal, Canadá, (2007)
4. Chinchuluun, A., Pardalos, P.M.: A survey of recent developments in multiobjective optimization. Ann. Oper. Res. **154**, 29–50 (2007)
5. Coello, C.A.C., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation). Springer, New York (2006)
6. Cormen, T.H., Leiserson, C.E., Rivest, R., Stein, C.: Introduction to Algorithms, 3rd edn. The MIT Press, Cambridge (2009)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
8. Ehrgott, M., Gandibleux, X.: A survey and annotated bibliography of multiobjective combinatorial optimization. OR-Spektrum **22**, 425–460 (2000)
9. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York (1979)
10. Gouveia, L., Magnanti, T.L.: Network flow models for designing diameter-constrained minimum-spanning and Steiner trees. Networks **41**, 159–173 (2003)
11. Gouveia, L., Simonetti, L., Uchoa, E.: Modeling hop-constrained and diameter-constrained minimum spanning tree problems as Steiner tree problems over layered graphs. Math. Program. **128**, 123–148 (2011)
12. Handler, G.Y.: Minimax location of a facility in an undirected graph. Transp. Sci. **7**, 287–293 (1978)
13. Hassin, R., Tamir, A.: On the minimum diameter spanning tree problem. Inf. Process. Lett. **53**(2), 109–111 (1995)
14. Ho, J.-M., Lee, D.T., Chang, C.-H., Wong, K.: Minimum diameter spanning trees and related problems. SIAM J. Comput. **20**(5), 987–997 (1991)
15. Kumar, R., Singh, P.K., Chakrabarti, P.P.: Multiobjective EA approach for improved quality of solutions for spanning tree problem. In: Proceedings International Conference on Evolutionary Multi-Criterion Optimization (EMO), Lecture Notes in Computer Science, pp. 811–825. Springer (2005)
16. Kumar, R., Singh, P.K.: On quality performance of heuristic and evolutionary algorithms for biobjective minimum spanning trees. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07, pp. 2259–2259. New York, NY, USA. ACM (2007)
17. Kumar, R., Rockett, P.I.: Improved sampling of the Pareto-front in multiobjective genetic optimizations by steady-state evolution: a Pareto converging genetic algorithm. Evol. Comput. **10**(3), 283–314 (2002)

18. Lemesre, J., Dhaenens, C., Talbi, E.G.: Parallel Partitioning Method (PPM): a new exact method to solve bi-objective problems. Comput. Oper. Res. **34**(8), 2450–2462 (2007)
19. Lucena, A., Ribeiro, C., Santos, A.C.: A hybrid heuristic for the diameter constrained minimum spanning tree problem. J. Global Optim. **46**, 363–381 (2010)
20. Marathe, M.V., Ravi, R., Sundaram, R., Ravi, S.S., Rosenkrantz, D.J., Hunt III, H.B.: Bicriteria network design problems. J. Algorithms **28**(1), 142–171 (1998)
21. Neumann, F., Wegener, I.: Minimum spanning trees made easier via multi-objective optimization. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, GECCO '05, pp. 763–769, New York, NY, USA. ACM (2005)
22. Noronha, T.F., Santos, A.C., Ribeiro, C.C.: Constraint programming for the diameter constrained minimum spanning tree problem. Electron. Notes Discrete Math. **30**, 93–98 (2008)
23. Noronha, T.F., Ribeiro, C.C., Santos, A.C.: Solving diameter-constrained minimum spanning tree problems by constraint programming. Int. Trans. Oper. Res. **17**(5), 653–665 (2010)
24. Raidl, G.R., Julstrom, B.A.: Greedy heuristics and an evolutionary algorithm for the bounded-diameter minimum spanning tree problem. In: Proceedings of the 18th ACM Symposium on Applied Computing, pp. 747–752. Melbourne (USA) (2003)
25. Raidl, G.R., Julstrom, B.A.: Edge sets: an effective evolutionary coding of spanning trees. IEEE Trans. Evol. Comput. **7**(3), 225–239 (2003)
26. Ramos, R.M., Alonso, S., Sicilia, J., Gonzalez, C.: The problem of the optimal biobjective spanning tree. Eur. J. Oper. Res. **111**(3), 617–628 (1998)
27. Requejo, C., Santos, E.: Greedy heuristics for the diameter-constrained minimum spanning tree problem. J. Math. Sci. **161**, 930–943 (2009)
28. Saha, S., Kumar, R.: Bounded-diameter MST instances with hybridization of multi-objective EA. Int. J. Comput. Appl. **18**(4), 17–25 (2011)
29. Santos, A.C., Lucena, A., Ribeiro, C.C.: Solving diameter constrained minimum spanning tree problem in dense graphs. Lect. Notes Comput. Sci. **3059**, 458–467 (2004)
30. Sourd, F., Spanjaard, O.: A multiobjective branch-and-bound framework: application to the biobjective spanning tree problem. INFORMS J. Comput. **20**(3), 472–484 (2008)
31. Steiner, S., Radzik, T.: Computing all efficient solutions of the biobjective minimum spanning tree problem. Comput. Oper. Res. **35**(1), 198–211 (2008)
32. Thomas, B.W., Chen, H., Campbell, A.M.: Network design for time-constrained delivery. Nav. Res. Logist. **55**, 493–515 (2008)
33. Veldhuizen, D.A.V., Lamont, G.B.: On measuring multiobjective evolutionary algorithm performance. In: Proceedings of the Congress on Evolutionary Computation, vol. **1**, pp. 204–211 (2000)
34. Wismer, D., Haimes, Y., Ladson, L.: On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE Trans. Syst. Man Cybern. **1**(3), 296–297 (1971)
35. Zhou, G., Gen, M.: Genetic algorithm approach on multi-criteria minimum spanning tree problem. Eur. J. Oper. Res. **114**, 141–152 (1999)
36. Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P.N., Zhang, Q.: Multiobjective evolutionary algorithms: a survey of the state of the art. Swarm Evol. Comput. **1**(1), 32–49 (2011)
37. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. Evol. Comput. **7**(2), 117132 (2003)
38. Zopounidis, C., Pardalos, P.M.: Handbook of Multicriteria Analysis, vol. 103. Springer, New York (2010)