RAIRO-Oper. Res. 50 (2016) 857–867
DOI: 10.1051/ro/2016060

RAIRO Operations Research
www.rairo-ro.org

AN EXACT METHOD TO GENERATE ALL NONDOMINATED SPANNING TREES

Asma Boumesbah[1] and Mohamed El-Amine Chergui[1]

Abstract. We describe an exact method to generate the nondominated set of the minimum spanning tree problem with at least two criteria. It is a separation and construction based method whose branching process is done with respect to edges belonging to at least two cycles of a given graph, inducing a step of constructing linear constraints that progressively break cycles while respecting the connectivity of the resulting graph. This has the effect of partitioning the initial graph into subgraphs, each of which corresponds to a discrete multi-objective linear program allowing to find the nondominated set of spanning trees. Randomly generated instances with more than two criteria are provided that show the efficiency of the method.

Mathematics Subject Classification. 90C10, 90C27, 90C29.

Received November 21, 2015. Accepted August 30, 2016.

1. Introduction

The minimum spanning tree problem (MST problem) is to find a least-cost spanning tree in an edge-weighted graph, it has several applications such as physical systems design, reducing data storage, cluster analysis and it is typically used in different sorts of distribution systems such as pipelines, transmission lines or in the design of leased-line telephone networks and other telecommunication problems. It also emerges as solutions to subproblems of a number of problems in combinatorial optimization.

The problem of multi-objective minimum spanning tree ($MOST$ problem) is one of the extended formulations of the MST problem. In this study, we describe an exact method to find the nondominated set and/or the efficient set of the $MOST$ problem in an undirected connected graph in which a cost-vector of dimension $r \geq 2$ is associated with each edge. A cost-vector Z of dimension r is said to dominate another cost-vector W of the same dimension, if Z is at least as good as W and Z is not equal to W. A spanning tree T is said efficient if it does not exist a spanning tree T' such that the cost-vector of T dominates the one of T'. Z is a Pareto optimal vector if it is nondominated by any other vector. The set of all nondominated spanning trees is known as the Pareto front. For more details on the multi-objective optimization concepts, see [16].

The $MOST$ problem is known to be NP-hard even for $r = 2$ [2,6] and although some methods developed for $r = 2$ were theoretically generalized for $r \geq 3$ [6,14], without any computational study.

Keywords. Minimum spanning tree – integer linear programming – multiple objective linear optimization – combinatorial optimization – branch and bound method.

[1] Faculty of Mathematics, USTHB, RECITS Laboratory, O2M team, Algiers, Algeria.
aboumesbah@usthb.dz; boumesbah-asma@hotmail.fr; mchergui@usthb.dz; mohamedelaminec@yahoo.com

Article published by EDP Sciences

In [15], the authors proposed an algorithm to solve the bi-objective case based on two phases. The first phase calculates two supported efficient solutions obtained by solving the single-objective spanning tree problem for each criterion. To generate the non-supported solutions, "k-best" algorithm [4] or the branch and bound method are used to solve a weighted sum program.

The authors of reference [3] aim to find all the nondominated solutions of the bi-objective minimum spanning tree problem by reducing the problem to a single-objective one. The method is based on the weighted sum function, which uses Kruskal's algorithm [9] to obtain the first supported spanning tree $T1$. To build a new spanning tree $T2$, the edges that are not in $T1$ are replaced by those which must leave $T1$, while preserving the non-dominance property.

Still in the bi-objective case, a branch and bound approach is given in reference [14]. This is mainly based on an estimation of all nondominated points by a given subproblem that takes advantage of the two-dimensional nature of the objective space. In fact, in this case it is easy to list the entire extreme supported points of a subproblem.

Corley's algorithm [5], is described to generate the efficient set of the $MOST$ problem. However, a counter example which shows that the algorithm is able to find spanning trees that are not efficient has been given by Hamacher and Ruhe [6].

For the multi-objective integer linear problems ($MOILP$), several methods are described to reach all non-dominated solutions but many of them are CPU time intensive using problems with fewer objectives (see [12]). Among these methods, three are reported, the first one [1] is an exact method to generate the nondominated set for $MOILP$ problems with r objective functions, $r \geq 2$. It is based on a branch and cut approach and uses a valid cut for the multi-objective case able to remove feasible solutions which are dominated. In the second one [11], the authors proposed a recursive algorithm to generate the full set of nondominated objective functions for the $MOILP$ problem. For any value of the bound lr corresponding to the objective function r, $r \geq 2$, each solution to the obtained problem yields a nondominated objective vector for the original $MOILP$ problem. In order to generate new solutions, the bound lr decreases (for minimization problems) in relation to the current nondominated solution until the corresponding integer linear program (ILP) is infeasible. The main contribution of the method cited in [12] is to improve the recursive algorithm developed in [11] for generating the set of all nondominated vectors. In fact, the new algorithm incorporates valuable informations that are not exploited in the further algorithm. This has the effect to avoid solving an important set of intermediate $ILPs$. The two methods [1, 12] are used as subroutines in our algorithm named MOST-Algorithm.

The aim of the present work is to provide a branch and bound based method to deal with the $MOST$ problem able to find the nondominated set with the criteria number $r \geq 2$. It is based on a two-step procedure starting by a separation step with respect to edges in common to at least two cycles of the given graph, yielding the step of constructing the problem constraints in the form of linear equalities and subcycle-elimination inequalities to ensure the non-existence of cycles. Each branch of the search tree induces a multi-objective linear program with binary variables ($MOLBP$) modeling the problem of the corresponding $MOST$ throughout this branch. Hence, the obtained $MOLBP$ programs can be solved by one of the two methods described in [1, 12]. The rest of the paper is organized as follows: we present in Section 1 the basic concepts and notations used in the document. In Section 2, the exact method to solve the $MOST$ problem is detailed, we describe the procedures of the algorithm in Section 3, and the main results are proved in Section 4. Then, the Section 5 is devoted to numerical experimentations performed on randomly generated instances using Matlab 2014a on an *hp* laptop, Intel (R) core (TM) i5-4200M CPU@ 2.50GHz, 8 GB Ram. Finally, Section 6 points out the conclusion of this study.

2. Definitions and notations

Given a connected and undirected simple graph $G = (V, E)$ of order n, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of vertices and $E = \{e_1, e_2, \ldots, e_m\}$ is a set of edges and each edge $e_i \in E$, $i = \overline{1, m}$, is valued by a cost-vector

$c_i = (c_{ik}),\ k = \overline{1,r},\ r \geq 2$. Let $T \subset E$ a spanning tree of G, the cost-vector of T is given by $C_k(T) = \sum_{e_i \in T} c_{ik},\ \ k = \overline{1,r}$. We note $C(T) = (C_k(T))_{k=\overline{1,r}}$ (see [5]).

We say that the vector $C(T)$ dominates another vector $C(T')$ if $C_k(T) \leq C_k(T')\ \ \forall k = \overline{1,r}$ and $C_k(T) < C_k(T')$ for at least one index $k \in \{1, \ldots, r\}$. A spanning tree T of G is efficient if there is no spanning tree T' of G such that $C(T')$ dominates $C(T)$.

In the multi-objective optimization programming, the ideal point is found by collecting each optimal objective values [16].

For the $MOST$ problem, the ideal point I has coordinates I_k such that:

$$I_k = C_k(T) = \min\left\{C_k(T) : T \textit{ spanning tree of } G\right\}, k = \overline{1,r}.$$

Throughout this paper $e2c$ denotes the edges type of the graph G belonging to at least two cycles.

Note the set $\overline{T} = \{e \in E : e \notin T\}$. Then, for each edge $e \in \overline{T}$, $T \cup \{e\}$ contains a unique cycle μ_e and $B = \{\mu_e, e \in \overline{T}\}$ is a cycle-basis of G. This means that the representative vectors $V(\mu_e)$ of edges of cycles $\mu_e \in B$ form a basis of the vector subspace of dimension $(m-n+1)$ of $\mathbb{R}^m$, where the coordinates of $V(\mu_e)$ are given by $V_j(\mu_e) = 1$ if edge $e_j \in \mu_e$ and 0 otherwise, for all $j = \overline{1,m}$. For more details see [10].

The mathematical model associated with the $MOST$ problem is written as follows [6]:

$$x_i = \begin{cases} 1 \ \text{ if the edge } \ e_i \in T,\ i = \overline{1,m} \\ 0 \ \text{ otherwise} \end{cases}$$

$$(P) \begin{cases} \text{Min}Z_1 = \displaystyle\sum_{i=1}^{m} c_{i1}x_i \\ \text{Min}Z_2 = \displaystyle\sum_{i=1}^{m} c_{i2}x_i \\ \qquad \vdots \\ \text{Min}Z_r = \displaystyle\sum_{i=1}^{m} c_{ir}x_i \\ \displaystyle\sum_{i=1}^{m} x_i = n-1 & (2.1) \\ \displaystyle\sum_{e_i \in E(S)} x_i \leq |S| - 1, \forall S \subset V, S \neq \emptyset & (2.2) \\ x_i \in \{0,1\}\ \ \forall i = \overline{1,m} \end{cases}$$

where $E(S)$ is the set of edges of the subgraph induced by $S,\ S \subset V$ [10].

Note that, in the model (P), the number of constraints (2.2) of subcycle-elimination inequalities is exponential and any attempt to solve it with an exact method is already useless in the single-objective case.

3. Principle of the method

We first start by recalling an important property in the graphs, namely an edge of a graph $G = (V, E)$ is either an isthmus (an edge whose deletion increases the number of connected components of G), or it belongs to a cycle of G [10]. Our method is based on this property in the sense that obtaining all the spanning trees of G is done by dissecting the edges of cycles of G, particularly those of type $e2c$. Indeed, the belonging or not of such edges to a spanning tree T prevents the creation of cycles in T. The proposed approach is a separation

and construction type in a structured search tree constructed and passed in depth-first approach. The separation is done on the $e2c$-edges of G and induces the construction step to describe constraints (2.2) of the program (P) which ensures the elimination of cycles relatively to $e2c$-edges. Each branch of the search tree is constructed by acting alternately on both separation and construction steps, which terminates at a leaf f without $e2c$-edges.

Hence, at each of such leaves f, we obtain a subgraph H with two possibilities. In the first one, the edges of H constitute a spanning tree reduced to the unique solution of the corresponding branch. The second possibility is that H contains only edge-disjoint cycles in which case the constraints eliminating all these cycles are added to the system of constraints previously established. The obtained set of constraints with binary variables and the criteria of program (P), constitutes a multi-objective linear program (P_f) that can be solved using any of the existing methods to generate the set $SMST_f$ of the efficient spanning trees and/or the SND_f set of nondominated cost-vectors associated with the leaf f. To provide computer simulation experiments, it is worth considering to adapt the faster of the two methods described in [1,12] to generate the SND set of nondominated solutions of the $MOST$ problem. We recall that the aim of this work is not to solve a $MOLBP$ program but to describe a general approach for the $MOST$ problem applying simultaneously properties of graph theory and linear programming.

To make faster our branch and bound based algorithm, an upper bound dominated set, SUB, is used. It contains some nondominated solutions found by optimizing weighted sums of the criteria (supported efficient solutions [16]) and potentially nondominated solutions, *i.e.*, solutions that are not dominated by any solution already known, are generated by an adapted Variable Neighborhood Search (VNS) algorithm [7]. Basically the VNS algorithm is based on systematically changing the neighborhood of generated solutions to avoid local minima and escape from the valleys which contain them. To adapt this idea to the $MOST$ problem, we consider a population of solutions, where the first population is randomly listed using Kruskal's algorithm. VNS is applied to each solution of the current potentially nondominated solutions in SUB set. Moreover, it is easy to calculate the mono-objective minimum spanning tree allowing to get the ideal point coordinates, which is used as a lower bound in the algorithm to fathom nodes of the search tree.

4. Procedures of the algorithm

In this section, the MOST-Algorithm steps are presented in details.

4.1. SUB Procedure

Input $G = (V, E)$: connected graph, costs of edges of G.
Output The initial SUB set of potentially nondominated solutions.
Step 1.1. Generate Ψ_1 a set of some supported spanning trees using a scalarization function.
Step 1.2. Generate Ψ_2 a set of potentially nondominated solutions using the adapted VNS algorithm. To do this, we consider only two types of neighborhood structures such that the neighbors of a given spanning tree T are obtained by changing k edges of T which do not belong to a same cycle of G, considering $k = 1$ or $k = 2$.
Step 1.3. The initial SUB set contains the nondominated solutions of the set $\Psi_1 \cup \Psi_2$.

4.2. Reduction Procedure

Input $G = (V, E)$: connected graph, costs of edges of G.
Output G': a partial graph of graph G, B: a cycle-basis of G', F: a set of $e2c$-edges of G'.
Step 2.1. Find a cycle-basis $B = \{\mu_1, \mu_2, \ldots, \mu_{m-n+1}\}$.
Step 2.1.1. If it exists an edge e belonging to a cycle μ of the basis B such that the cost-vector of e is dominated by all those of the edges of μ, then delete e from the graph G to obtain a partial graph G' (see Prop. 5.1).
Step 2.1.2. If it exists an edge e belonging to a set Δ of cycles of B such that the cost-vector of e dominates all those of the edges of Δ, then the edge e belongs to all nondominated spanning trees of $MOST$ problem (see Prop. 5.2).

Step 2.2. Determine a new basis B.
Step 2.3. Find the set $F = \{e_0 \in E / \exists \mu_l \text{ and } \mu_{l'} \in B; e_0 \in \mu_l \cap \mu_{l'}\}$.

4.3. Separation and Construction Procedure

Input G', B, F and SUB.
Output A set of linear constraints modeling the spanning trees of the branch that terminates at the current leaf f.
Step 3.1.
a- If $B = \{\mu_1, \mu_2, \ldots, \mu_{m-n+1}\}$ and $F = \emptyset$, we are in presence of a leaf f, then add the set of constraints to break all edges-disjoints cycles in B:

$$\begin{cases} \sum\limits_{e_j \in \mu_i} x_j \leq |\mu_i| - 1, \forall i = \overline{1, m-n+1} & (4.1) \\ \sum\limits_{e_i \in E} x_i = n - 1 & (4.2) \end{cases}$$

b- If $B = \emptyset$, $F = \emptyset$ and $(\sum_{e_i \in E} x_i = n - 1)$, we are in presence of a leaf f, then a spanning tree is obtained. Fathom this leaf.
c- If $F \neq \emptyset$ then, go to Step 3.2.
Step 3.2. Select an edge $e_i \in F$ and create two new nodes.
Step 3.2.1. In the first node, add the following constraint to the set of constraints previously imposed at the parent node:

$$x_i = 1 \tag{4.3}$$

a- If the edge e_i creates a cycle with the set of already fixed edges, then this node is fathomed.
b- Else, add the following constraints to break cycles μ_l and $\mu_{l\prime}$ containing the edge e_i:

$$\begin{cases} \sum\limits_{e_j \in \mu_l} x_j \leq |\mu_l| - 1 & (4.4) \\ \sum\limits_{e_j \in \mu_{l\prime}} x_j \leq |\mu_{l\prime}| - 1 & (4.5) \\ x_j \in \{0,1\} \quad j = \overline{1, |\mu_l| + |\mu_{l\prime}| - 1} \end{cases}$$

Step 3.2.2. In the second node, add the constraint to the set of constraints previously imposed at the parent node:

$$x_i = 0 \tag{4.6}$$

Consider the graph $G' = G \setminus \{e_i\}$ and go to Reduction Procedure.
Go to Step 3.1.

The next step corresponds to a leaf f involving a multi-objective linear program with binary variables (P_f) written in the general form as given below.

4.4. Evaluation Procedure

Input A linear binary variables program with multiple objectives.
Output SND_f the set of the nondominated cost-vectors and/or $SMST_f$ the associated efficient spanning trees set.

Step 4.1 Solve the following (P_f) program. This is done using the method [12] for the programs with number of criteria less or equal to four. For a greater criteria number, the method cited in [1] is used.

$$(P_f)\begin{cases} \text{Min}Z_i = \sum_{j=1}^{m} c_{ji}x_j \quad ; i = \overline{1,r} \\ \sum_{e_j \in \mu_l} x_j \leq |\mu_l| - 1; l \in L \\ \sum_{e_j \in E} x_j = n - 1 \\ x_p = 1; p \in M \\ x_q = 0; q \in N \end{cases}$$

where along the branch of leaf f, M denotes the set of indices of edges belonging to all efficient trees, N denotes the set of indices of edges excluded from all efficient trees and L is the set of indices of cycles of the constructed basis.
Step 4.2. Return SND_f and/or $SMST_f$ sets.

In the following, we present the MOST-Algorithm to solve the $MOST$ problem. This main algorithm calls all the procedures presented earlier to obtain the set of all nondominnated spanning trees.

MOST-Algorithm
Input $G = (V, E)$: connected graph, C: cost-vector of edges of G.
Output SND: list of the nondominated solutions of $MOST$ problem.
Step 5.1. Execute **SUB Procedure**
Step 5.2. Execute **Reduction Procedure**
Step 5.3. While there still exist unfathomed nodes in the search tree, do:
Step 5.3.1. Choose a node according to the depth-first strategy.
Step 5.3.2 Determine the coordinates of the ideal point I corresponding to this node.
Step 5.3.3. Compare I with the elements of the SUB set:
Step 5.3.3.1. If I is dominated by an element z of the SUB set then fathom this node and go to Step 5.3.
Step 5.3.3.2. Else, go to Step 5.3.4.
Step 5.3.4. Execute **Separation and Constuction Procedure**
Step 5.3.4.1. If at the current leaf f a spanning tree T_f is obtained, then: $SND_f := C(T_f)$. Fathom this node and go to Step 5.3.
Step 5.3.4.2. Evaluation Procedure
For all costs-vector C_T in SND_f, do:
a- If C_T is not dominated by z, $\forall z \in SUB$, then $SUB := SUB \cup \{C_T\}$.
b- If $\exists z \in SUB/z$ is dominated by C_T, then $SUB := SUB \setminus \{z\} \cup \{C_T\}$. Fathom this node and go to Step 5.3.
Step 5.4. Terminate, $SND := SUB$.

5. Main results

In this section the following results are proved to justify the convergence of the MOST-Algorithm.

Let $G = (V, E)$ be a connected and undirected graph of order n and size m. The following propositions allow to highlight particular edges of the graph G whose the belonging or not to nondominated spanning trees is proved according to the Reduction Procedure of the MOST-Algorithm. Note that the following Proposition 5.1 has been already used in [14].

Proposition 5.1. *Let μ be a cycle of G and e an edge of G, $e \in \mu$. If all cost-vectors of other edges of the cycle μ dominate the one of the edge e, then e can not belong to any nondominated spanning tree.*

Proof. Let T be a spanning tree and $e \in \mu$, μ a cycle of G such that all cost-vectors of the other edges of cycle μ dominate the one of the edge e and assume that $e \in T$. Let $f \in \mu$ and $f \notin T$, then $T' = T \cup \{f\} \setminus \{e\}$ is a spanning tree since $e \in T \cup \mu$ and $f \in \mu$ but $f \notin T$. Then, $C_k(T') = C_k(T) + c_{fk} - c_{ek}$, $\forall k = \overline{1,r}$. Hence,

$C_k(T') \leq C_k(T), \quad \forall k = \overline{1,r}$, with at least one strict inequality, because $c_{fk} - c_{ek} \leq 0, \quad \forall k = \overline{1,r}$, with at least one strict inequality as the cost-vector of f dominates that of e. Then the cost-vector of T' dominates the one of T. □

Proposition 5.2. *Let e be an edge of G common to a set Δ of cycles of G, and assume that the cost-vector of e dominates all other cost-vectors of the edges of cycles in Δ, then the edge e belongs to all nondominated spanning trees of G.*

Proof. We denote $E(\Delta)$ the set of all edges in Δ. If there exists a nondominated spanning tree T which does not contain the edge e, then $T' = T \cup \{e\} \setminus \{f\}, \forall f \in E(\Delta)$ is a spanning tree whose cost-vector dominates the one of the spanning tree T because the edge e dominates the edge $f, \forall f \in E(\Delta)$. □

Let B a basis of cycles of a given graph, the way to breaking the cycles of B depends on the existence or not of $e2c$-edges in order to associate the specific linear constraints in the Separation and Construction Procedure. To justify the non existence of $e2c$-edges, we prove the following theorem.

Theorem 5.3. *If G admits a cycle-basis B such that all cycles are edge-disjoint, then there is no other basis of cycles B' whose two cycles have common edges.*

Proof. Suppose that there is a cycle basis B' such that there exist two cycles μ_1 and μ_2 in B', with μ_1 and μ_2 having at least one common edge and $\mu_1 \neq \mu_2$.
Let $B = \{\sigma_i, i = \overline{1, m-n+1}\}$ another cycle-basis of G.
As $\mu_1 \cap \mu_2 \neq \emptyset$, then $\exists e_j \in E$ such that $e_j \in \mu_1 \cap \mu_2$. Otherwise, in the basis B the representative vectors of cycles μ_1 and μ_2 are written in a unique way according to those of cycles of B:

$$\begin{cases} V(\mu_1) = \sum_{i=1}^{m-n+1} \alpha_i V(\sigma_i) \\ V(\mu_2) = \sum_{i=1}^{m-n+1} \beta_i V(\sigma_i) \end{cases}$$

such that: $V_j(\mu_k) = 1 \; if \; e_j \in \mu_k$, 0 otherwise, $k = 1, 2$ and $\quad \alpha_i, \beta_i \in \{-1, 0, 1\}$.

On the other hand, as the cycles $\sigma_i, i = \overline{1, m-n+1}$ are elementary edge-disjoint, there is only one cycle σ_{i_0} containing the edge e_j proving that $\sigma_{i_0} = \mu_1$ and $\sigma_{i_0} = \mu_2$. Contradiction with $\mu_1 \neq \mu_2$. □

The following theorem proves the convergence of the MOST-Algorithm.

Theorem 5.4. *The MOST-Algorithm is able to construct all nondominated spanning trees in a finite number of iterations.*

Proof. It is obvious that the separation step of the Separation and Construction Procedure ensures the independence of the problems obtained at the nodes of the search tree and hence, the non-redundancy of the spanning trees found, while the construction step puts in evidence the constraints that the edges of a given cycle μ of G should not all together be taken, otherwise the obtained solutions would not be spanning trees. This leads to find all linear constraints of the $MOST$ problem at each leaf f of the search tree, allowing to the Evaluation Procedure to determine the nondominated set SND_f, then the set of potentially nondominated spanning trees SUB is updated accordingly. The number of created leaves being finished, therefore the set of all nondominated spanning trees SND is obtained in a finite number of iterations. □

Example

Consider the graph $G = (V, E)$ where $V = \{1, 2, 3, 4, 5, 6\}$ and $E = \{12, 13, 15, 23, 34, 36, 45, 56\}$. With each edge is associated three costs as shown in the following matrix of costs:

$$C = \begin{pmatrix} 0 & 2 & 1 & 2 & 4 & 0 & 1 & 3 \\ 3 & 3 & 2 & 2 & 3 & 2 & 1 & 1 \\ 1 & 0 & 3 & 2 & 4 & 0 & 1 & 2 \end{pmatrix}$$

SUB Procedure

$$ST_1 = \{12, 13, 15, 45, 36\}, Z(ST_1) = (4, 11, 5).$$
$$ST_2 = \{12, 13, 45, 65, 36\}, Z(ST_2) = (6, 10, 4),$$
$$ST_3 = \{12, 15, 45, 65, 36\}, Z(ST_3) = (5, 9, 7),$$
$$ST_4 = \{15, 23, 45, 56, 36\}, Z(ST_4) = (7, 8, 8).$$
$$SUB = \{Z(ST_1), Z(ST_2), Z(ST_3), Z(ST_4)\}.$$

Reduction Procedure
Let the cycle-basis $B = \{\mu_1, \mu_2, \mu_3\}$ with $\mu_1 = \{13, 36, 65, 51\}, \mu_2 = \{12, 23, 31\}$ *and* $\mu_3 = \{43, 36, 65, 54\}$.

- In the cycle μ_3 the cost-vector of the edge 34 is dominated by the other costs-vectors of all edges in μ_3. According to Proposition 5.1, this edge is removed from the graph and $E := E \setminus \{34\}$.
- Determine then a new cycle-basis and find the set $F = \{ij \in E \setminus \exists\ \mu_l \text{ and } \mu_{l'} \in B;\ ij \in \mu_l \cap \mu_{l'}\}$.

Let $B = \{\mu_1, \mu_2\}$ with $F = \{13\}$.

Separation and Construction Procedure

$$x_{ij} = \begin{cases} 1 & if \quad the \quad edge \quad ij \in T, \quad i, j = \overline{1, n} \\ 0 & otherwise \end{cases}$$

The separation is done with respect to the edge 13 creating two nodes. At the node 1 we deduce the following system:

$$(S_1) \begin{cases} x_{13} = 1 \\ x_{36} + x_{65} + x_{15} \leq 2 \\ x_{32} + x_{21} \leq 1 \\ x_{12} + x_{15} + x_{23} + x_{36} + x_{45} + x_{65} = 4 \\ x_{12}, x_{13}, x_{15}, x_{23}, x_{36}, x_{45}, x_{65} \in \{0, 1\} \end{cases}$$

Evaluation Procedure (at node 1)
$F := \emptyset$, then the node 1 is a leaf. By using the method described in [12] for the resolution of multi-objective linear program with binary variables (P_1) whose systems of constraints is (S_1), it returns all efficient spanning trees associated with this branch of the search tree:

$$T_1 = \{12, 13, 45, 65, 36\}, Z(T_1) = (6, 10, 4),$$
$$T_2 = \{13, 23, 45, 65, 36\}, Z(T_2) = (8, 9, 5),$$
$$T_3 = \{12, 13, 15, 45, 36\}, Z(T_3) = (4, 11, 5).$$
$$SND_{f_1} = \{Z(T_1), Z(T_2), Z(T_3)\}.$$
$$SUB = SUB \cup Z(ST_2).$$

Reduction Procedure
At the node 2 of the search tree, the edge 13 is deleted and the graph $E := E \setminus \{13\}$. The coordinates of the ideal point I corresponding to this node: $Z(I) = (4, 8, 6)$ which is not dominated by any solutions found previously in the first node, therefore the second node is not fathomed.
The new basis $B = \mu_4$ was obtained with $\mu_4 = \{12, 23, 36, 65, 51\}$.

Separation and Construction Procedure
The corresponding system of linear constraints is:

$$(S_2)\begin{cases} x_{13} = 0 \\ x_{12} + x_{23} + x_{36} + x_{65} + x_{51} \leq 4 \\ x_{12} + x_{15} + x_{23} + x_{36} + x_{45} + x_{65} = 5 \\ x_{12}, x_{13}, x_{15}, x_{23}, x_{36}, x_{45}, x_{65} \in \{0, 1\} \end{cases}$$

Evaluation Procedure (at node 2)
In this leaf, The method described in [12] is used to solve the multi-objective linear program with binary variables (P_2) associated with (S_2), all efficient spanning trees associated with this branch are:

$$\begin{aligned} T_4 &= \{12, 23, 45, 65, 36\}, Z(T_4) = (6, 9, 6), \\ T_5 &= \{12, 15, 45, 65, 36\}, Z(T_5) = (5, 9, 7), \\ T_6 &= \{12, 23, 15, 45, 36\}, Z(T_6) = (4, 10, 7), \\ T_7 &= \{15, 23, 45, 56, 36\}, Z(T_7) = (7, 8, 8). \\ SND_{f_2} &= \{Z(T_4), Z(T_5), Z(T_6), Z(T_7)\}. \\ SUB &= SUB \cup \{Z(T_4), Z(T_6)\}. \end{aligned}$$

Thus, the final efficient set of spanning trees of G is: $SMST = \{ST_1, ST_2, ST_3, ST_4, T_2, T_4, T_6\}$ and the associated set of nondominated solutions is: $SND/SND := SUB$.

6. Experiment results

The experimental study is performed under Matlab 2014a on an hp laptop, Intel core i5, 8GB of Ram, on randomly generated instances. In this case, all the generated instances consider the cost-vectors of dimensions 3, 4, 5, which are uniformly distributed in the interval $[-10, 50]$. The graphs are randomly generated as described by Erdos−Renyi in [8] who proposed to the user to fix the number of vertices of the graph and the probability that an edge exists between two vertices. For this fact, we have found useful to group in each line of the Table 1 the average results for twenty (20) graphs for which the number of edges ranges in a same interval of length not exceeding ten (10).

In Table 1, we recorded the average, the minimum and the maximum number of nondominated solutions *nbSND*, as well as the CPU time in seconds, the *Cub* number which represents the run time spent to generate the initial set SUB and the average number of leaves (*nbL*).
For instances with three and four criteria, the best results are obtained using the method in [12] at each leave, whereas all the other results are from the method in [1] which is faster with increasing number of criteria.

The results clearly show the increase of the CPU time in relation to the number of edges and that of leaves (*nbL*). However, the number of criteria does not seem greatly act on this CPU time and the exact evaluation of the ideal point in each node of the search tree brought an improvement to the method since the number of fathomed nodes represents an average of 55.63 % of the total number of created leaves (*nbL*).

Moreover, the set SUB considered as an upper bound for the SND set along the steps of the MOST-Algorithm, evolves in a dynamic way and is updated in a polynomial time. Indeed, the set SND contain potentially nondominated solutions that we update during the search. In particular, any element of this set could be deleted if it is dominated by a new solution recently met.

The running time of MOST-Algorithm decreases with an increasing number *nb e2c* of edge-disjoint cycles in G. Indeed, graphs with low number *nb e2c* have also been considered. In this case the results reported in Table 2 show that the MOST-Algorithm is faster making larger dimensions of graphs tractable.

Along our experimentation, the calculation is interrupted for instances requiring more than 2 h of computing.

TABLE 1. Experiment results.

r	n	m	$nbSND$			$CPU(s)$			Cub	nbL
			Avg	Min	Max	Avg	Min	Max	Avg	Avg
3	20	[35, 45]	38, 1	18	102	30, 7	10, 4	87, 9	0, 04	301, 2
3	20	[46, 55]	67	35	105	115, 9	45, 2	216, 8	0, 1	595, 4
3	30	[45, 55]	121, 3	51	186	337, 1	83	858, 2	0, 08	2240, 8
3	30	[56, 65]	273, 2	87	403	1203	703, 2	2301, 7	0, 5	2603
3	50	[65, 75]	881, 6	107	1025	2489, 9	1833, 2	3581, 7	1, 2	4843, 2
4	20	[35, 45]	102	66	222	44, 2	28, 2	178	1, 17	356, 4
4	20	[46, 55]	104, 3	44	130	186, 7	70, 3	227, 7	1, 3	660
4	30	[45, 55]	188, 5	63	305	557, 4	112, 2	1038, 4	2, 6	2452, 6
4	30	[56, 65]	320, 3	94	690	1723, 9	921, 1	3074	3, 02	3008, 7
4	50	[65, 75]	1060, 2	230	2003	3030, 68	2100, 4	3864	4, 6	5036, 4
5	20	[35, 45]	246	87	440	88, 2	47, 2	206	4, 8	489, 5
5	20	[46, 55]	274, 8	105	473	294, 5	161, 6	924, 9	5, 01	1090, 8
5	30	[46, 55]	758, 3	123	1053	1033, 4	452, 6	2785	5, 21	2821, 3
5	30	[55, 65]	930, 3	202	2066	2034, 7	1132, 7	3520, 3	7, 09	3140, 6
5	50	[65, 75]	1300, 4	314	2135	3423, 1	2306, 1	4037, 8	7, 9	5453, 5

TABLE 2. Results in average for graphs having a small number of $e2c$-edges.

r	n	m	$nb\ e2c$	$CPU(s)$	$nb\ SND$
3	100	130	3	47, 3	148
3	300	340	4	956, 24	566
3	400	445	5	1030, 32	759
4	100	130	3	1404, 38	768
4	300	340	3	1521, 12	1095
4	400	445	5	2234, 81	1155

7. CONCLUSION

The main idea of the presented exact method is based on a branching process on common edges with at least two cycles of a given graph. This generates a procedure of constructing constraints which break cycles while keeping the connectivity of the graph. These constraints become easy to enumerate after the branching process. We want to emphasize that at each leaf of the search tree the obtained graph has a particular structure; it contains only edge-disjoint cycles, otherwise it is a spanning tree. Therefore if the first basis of the initial graph is edge-disjoint cycles, then the algorithm creates only one leaf and only one bivalent linear multi-objective program is solved which induces the faster execution time. Else at each leaf of an arbitrary branch of the search tree, a bivalent linear multi-objective program of the minimum spanning tree problem is solved according to the constraints induced along this branch. Hence, the obtained zero-one sparse matrix of constraints is much smaller than the exponentially dimension of the initial one leading to a faster resolution. On the other hand, computing the ideal point coordinates is an easy problem and it allowed us to not visit areas that do not contain nondominated spanning trees. Indeed, in average more than 50 % of nodes of the search tree are discarded and this explains why the method is able to treat thousands of leaves in an acceptable time. Only the use of a dedicated ($MOLBP$) exact method will enable the fullest implementation of computer simulations and assessment of results deservedly. Note that the aim of this work is not to make a comparative study with existing methods dedicated to the bi-objective spanning tree problem but rather highlight a new method able to solve this problem with more than two objective functions. As future studies, the construction of nondominated spanning trees based on graphs tools seems to be worth to investigate.

References

[1] M. Abbas, M.E.-A. Chergui and M. Ait Mehdi, Efficient cuts for generating the nondominated vectors for multiobjective integer linear programming. *Int. J. Math. Oper. Res.* **4** (2012).

[2] P.M. Camerini, G. Galbiati and F. Maffioli, The complexity of weighted multi-constrained spanning tree problems. *Colloquium on the Theory of Algorithms*. Edited by L. Lovàsz. North-Holland, Amsterdam (1984) 53–101.

[3] C.G. Da Silva and J.C.N. Clìmaco, A note on the computation of ordered supported nondominated solutions in the bi-criteria minimum spanning tree problems. *J. Telecomm. Inform. Techn.* (2007) 11–15.

[4] J.C.N. Clìmaco, E and A. Martins, bicriterion shortest path algorithm. *Eur. J. Oper. Res.* **11** (1982) 399–404.

[5] H.W. Corley, Efficient spanning trees, Craveirinha and M. Pascoal, Multicriteria routing models in telecommunication networks-overview and a case study. *J. Optim. Theory Appl.* **45** (1985) 481–485.

[6] H.W. Hamacher and G. Ruhe, On spanning tree problems with multiple objectives. *Annal. Oper. Res.* **52** (1994) 209–230.

[7] P. Hansen and N. Mladenović, Variable neighborhood search. *Comput. Oper. Res.* **24** (1997) 1097–1100.

[8] P. Erdös and A. Rényi, On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.* **5** (1960) 17–61.

[9] J. B. Kruskal, On the Shortest Spanning Subtree of a Graph and the Travelling Salesman Problem. *Proc. Amer. Math. Soc.* **7** (1956) 48–50.

[10] C.W. Marshall, *Applied graph theory*. Wiley-Interscience (1971).

[11] M. Özlen and M. Azizoğlu, Multi-objective integer programming, A general approach for generating all nondominated solutions. *Eur. J. Oper. Res.* **199** (2009) 25–35.

[12] M. Özlen, B.A. Burton and C.A.G. MacRae, Multi-objective integer programming: An Improved Recursive Algorithm. *J. Optim. Theory Appl.* **160** (2014) 470–482.

[13] R.C. Prim, Shortest Connection Networks and Some Generalizations. *Bell Syst. Tech. J.* **36** (1957) 1389–1401.

[14] F. Sourd and O. Spanjaard, multi-objective branch-and-bound framework. Application to the bi-objective spanning tree problem. *INFORMS J. Comput.* **20** (2008) 472–484.

[15] S. Steiner and T. Radzik, Computing all efficient solutions of the biobjective minimum spanning tree problem. *Comput. Oper. Res.* **35** (2008) 198–211.

[16] R.E. Steuer, Multiple criteria optimization: theory, computation, and application. *Wiley Series Prob. Math. Stat. Appl.* Wiley (1986).