Computational Optimization and Applications (2021) 78:491–528
https://doi.org/10.1007/s10589-020-00251-6

Finding multi-objective supported efficient spanning trees

Pedro Correia[1] · **Luís Paquete**[1] · **José Rui Figueira**[2]

Received: 13 March 2019 / Accepted: 2 December 2020 / Published online: 25 January 2021

Abstract
This article introduces a new algorithm for computing the set of supported non-dominated points in the objective space and all the corresponding efficient solutions in the decision space for the multi-objective spanning tree (MOST) problem. This algorithm is based on the connectedness property of the set of efficient supported solutions and uses a decomposition of the weight set in the weighting space defined for a parametric version of the MOST problem. This decomposition is performed through a space reduction approach until an indifference region for each supported non-dominated point is obtained. An adjacency relation defined in the decision space is used to compute all the supported efficient spanning trees associated to the same non-dominated supported point as well as to define the indifference region of the next points. An in-depth computational analysis of this approach for different types of networks with three objectives is also presented.

Keywords Multi-objective optimization · Weight-set decomposition · Minimum spanning tree · Neighborhood search

1 Introduction

The *minimum spanning tree* (MST) problem is a combinatorial optimization problem with many real-world applications [2, 37, 38], which are present in almost our daily life aspects (including our interactions with respect to social networks, computer networks, telecommunications networks, transportation networks, water

✉ Luís Paquete
paquete@dei.uc.pt

Pedro Correia
pamc@dei.uc.pt

José Rui Figueira
figueira@tecnico.ulisboa.pt

[1] CISUC, Department of Informatics Engineering, University of Coimbra, Coimbra, Portugal

[2] CEG-IST, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal

supply networks, electrical grids, and much more). Other practical applications of minimum spanning tree include clustering analysis, taxonomy, circuits design, image registration and segmentation, handwriting recognition, automatic speech recognition, socio-geographic applications, surface homogeneity tests, classification problems, amongst others [8, 13, 24, 26, 30, 32].

Real-life problems deal very often with conflicting objectives. Consider, for example, a broadcasting network to send files to several connected endpoints. It is natural that the best and most reliable network links have a better performance when sending files, but simultaneously are more expensive if they are used.

Moving from single objective spanning tree problems to *multiple objective spanning tree* (MOST) problems means to replace the concept of optimal spanning tree (solution) by that of *efficient spanning tree* in the decision space. The image of an efficient solution in the objective space is called *non-dominated point*. The search for the entire set of efficient spanning trees and/or the corresponding non-dominated points is a fundamental problem in multi-objective combinatorial optimization [10].

A relevant subset of the set of efficient spanning trees is called the set of *supported spanning trees*. These spanning trees are optimal for the weighted sum of several objective functions. It is easier to find a supported efficient spanning tree than an efficient spanning tree in general; the latter is an NP-hard problem whereas the former can be solved with classical algorithms, such as Kruskal algorithm, in polynomial amount of time. For this reason, this subset of spanning trees are usually found in a pre-processing phase and used for pruning within implicit enumeration strategies.

For the bi-objective spanning tree problem, Ahuja et al. [2] devised an algorithm to obtain the set of supported spanning trees. The algorithm exploits the relations between supported spanning trees and *adjacent* trees. Two trees are adjacent if and only if they share all but one edge. The algorithm starts by calculating a lexicographical optimal spanning tree w.r.t. one objective function, and then, by searching amongst its adjacent trees, it calculates the weightings for which that tree is optimal for the scalarized version of MOST with respect to the weighted sum (so called parametric problem). To find those weightings the algorithm needs to find, among all the neighboring trees, the one that has a better weighted sum for a larger value of the objective function considered initially. Such tree is the next supported spanning tree. The process is repeated until the lexicographical optimal solution w.r.t. the second objective function is found.

Seipp proposed an algorithm to calculate a subset of the supported spanning trees (the *extreme* supported spanning trees) by performing a division on the weight set for the tri-objective spanning tree problem [33]. In his work, the weight set is intersected with a polyline for each pair of edges. Every sub-space generated leads to a subset of the weighted set. For each weight set division, a parametric optimal spanning tree problem is solved. Sub-weight sets are merged whenever they lead to the same optimal spanning tree. Although partially described, the author does not give implementation details and do not report any experimental analysis. It is not clear which data structures could be used to store the decomposed weight set and how could the algorithm be implemented efficiently.

In this article, we propose an algorithm that finds the whole set of supported efficient solutions for the MOST problem. Our approach is based on the identification of the relations among supported efficient solutions and their immediate neighborhood solutions in terms of adjacency both in the decision and in the objective space. Similarly to the bi-objective version described in [2], we calculate the weight set region for which every supported solution is optimal for the parametric problem. Such a region in the general MOST problem is a *polytope*, which we refer to as an *indifference region*. This algorithm starts by finding a lexicographical solution by calculating the single objective minimum spanning tree for a problem with edges ordered lexicographically. The remaining supported spanning trees of the problem and the indifference region associated to each spanning tree are calculated by performing neighborhood search starting from a lexicographical optimal solution.

This article is organized as follows. Section 2 is devoted to concepts, definitions, and notation. Section 3 deals with the presentation of the proposed algorithm. Section 4 provides an illustrative example. Section 5 presents a computational study. Finally, Sect. 6 presents some conclusions and future work.

2 Concepts, definitions, and notation

This section introduces the minimum spanning tree problem along with some fundamental results, the presentation of its multi-objective variant, the weighted-sum spanning tree problem, and some algorithmic aspects for the three-objective spanning tree problem.

2.1 The minimum spanning tree problem

Consider an undirected and connected *valued graph*, $G = (V, E, c)$, where V is the *set of vertices* and $n = |V|$, E is the *set of edges* where $m = |E|$, and c is a *set of values*, one *per* edge. An *edge* connecting vertex i to vertex j is denoted by $\{i, j\}$ and $c_{ij} \in \mathbb{R}_+$ denotes the *value* of such an edge. The *minimum spanning tree* (*MST*) problem consists of finding a spanning tree in G whose sum of edge values is as small as possible. Let T^* denote an optimal spanning tree.

Two equivalent optimality conditions are considered (see pages 518–519 in [2]).

Proposition 1 (MST optimality condititions)

(a) *Cut-based optimality condition*: *For all edges* $\{i, j\} \in T^*$, $c_{ij} \leqslant c_{k\ell}$ *for all edges* $\{k, \ell\}$ *belonging to the cut formed by removing edge* $\{i, j\}$ *from* T^*.
(b) *Path-based optimality condition*: *For all edges* $\{k, \ell\} \notin T^*$, $c_{ij} \leqslant c_{k\ell}$ *for all edges* $\{i, j\}$ *belonging to the path of* T^* *from vertex* k *to vertex* ℓ.

These conditions are important for devising algorithms and for proving their correctness (see [2]). The asymptotically fastest algorithms in the literature are the approach described in Karger et al. [20], which is a randomized algorithm that

solves the minimum spanning tree in expected linear time, and in Chazelle [7], which consists of a deterministic algorithm that is able to solve the problem in $\mathcal{O}(m\alpha(m,n))$ time, where $\alpha(m,n)$ is the inverse Ackermann function [1]. The implementation of these very efficient algorithms is rather complex. In general, variants of Kruskal's and Prim's algorithms are considered [22, 26]. The first runs in $\mathcal{O}(m \log n)$ while the second runs in $\mathcal{O}(n^2)$. Let us remark that when using a Fibonacci heap, Prim's algorithm runs in $\mathcal{O}(n + m \log m)$.

Let $\mathcal{T}$ be the set of all feasible spanning trees in G and let function $z : \mathcal{T} \to \mathbb{R}$. The set $\mathcal{T} \subset E^{n-1}$ is the *feasible region in the decision space*. The cardinality of $\mathcal{T}$ for a complete graph is given by the Cayley's formula [6], i.e., $|\mathcal{T}| = n^{n-2}$.

The *MST* problem can also be stated as follows: $\min z(T)$, subject to $T \in \mathcal{T}$, where

$$z(T) = \sum_{\{i,j\} \in T} c_{ij}$$

is the objective function to be minimized.

The concept of *adjacent* spanning trees (see page 538 in [2]) is important in the context of this work.

Definition 1 (*Adjacency in the decision space*) Consider two feasible spanning trees, T', $T'' \in \mathcal{T}$. They are said to be *adjacent* if both share all, but an edge.

The edge swapping that allows to move from T' to T'' is called an *elementary move* and it is composed of two edges $\{i,j\}$, and $\{k,\ell\}$, where $\{i,j\} \in T'$, and $\{k,\ell\} \in T''$; we denote it by $s^{T',T''} = \langle \{i,j\} \leftrightarrow \{k,\ell\} \rangle$. All the trees that are reachable by elementary moves from T' are called the *immediate neighbors* of T'.

The adjacency relation in Definition 1 allows to form an adjacency graph [11] composed of adjacent spanning trees. For a given set of such spanning trees, if the graph is connected, we say that the set is connected. Let $\mathcal{T}^*$ denote the set containing all MSTs. Such set is connected [31].

Proposition 2 (Connectedness of all **MSTs** [31]) *Let T' and T'' denote two minimum spanning trees. There is a sequence of elementary moves yielding T'' from T' such that all spanning trees constructed in this sequence are minimum spanning trees.*

We remark that it is easy to see, from linear programming, that the set of all MSTs is connected. We can associate a $\{0,1\}$ variable, x_{ij}, with each edge $\{i,j\} \in E$. This variable x_{ij} is equal to 1 if edge $\{i,j\}$ belongs to a tree; otherwise, it assumes the value 0. A tree T can be represented by a solution or vector x with components 1 for the edges in T and 0 for the edges in $E \backslash T$. Let X denote the set of all feasible solutions (trees) and $conv(X)$ the convex hull of X. Observe that $conv(X)$ is a polytope with at most $(n-1)$ dimensions, where all the extreme points are with $\{0,1\}$ components. It means that it is possible to relax variable x_{ij} to be nonnegative since its value will only be either 0 or 1. Now, when optimizing a linear function over $conv(X)$, all the optimal solutions can be computed. When there is no degeneracy (in which case there is no change in the basic variables with $x_{ij} = 1$) a

non-basic variable $x_{k\ell}$ will replace a basic variable $x_{ij} = 1$. For the computation of all alternative optimal solutions, see Chapter 4 in Steuer [36]. For the degenerate pivoting, the value of the objective function does not change. As for the non-degenerate pivoting if the optimal solutions were not connected the entering variable would degrade the value of the objective function, which is not possible. In case the value of the objective function improves, the previous solution was not at the optimum, which is also not possible. Hence, we can prove that the optimal solutions are connected in the sense of linear programming. Note that there is an equivalency between the linear programming adjacency and the combinatorial concept of adjacency (see page 95 in Gorski [16]).

A naïve algorithm to find all *MSTs* based on Proposition 2 runs in time $\mathcal{O}(mn|\mathcal{T}^*|)$ since for each minimum spanning tree, all elementary moves from that tree to another tree are required to be tested. Eppstein proposed a faster algorithm to generate $\mathcal{T}^*$ that runs in $\mathcal{O}(m + n\log n + |\mathcal{T}^*|)$ [12]. Eppstein's algorithm is based on the transformation of the valued graph, G, into a non-valued auxiliary graph, $\hat{G}$. This transformation is performed such that there is a one-for-one correspondence between each spanning tree of $\hat{G}$ and each minimum spanning tree of G. Such transformation is performed by first calculating an *MST* of G, T^*, and then performing a succession of modifications to the edges of $G \backslash T^*$ based on the weights of such edges and the weights of the edges of T^*. All the spanning trees of $\hat{G}$ are listed using an algorithm proposed in Kapoor and Ramesh [19].

2.2 The multi-objective spanning tree problem

Let $c_{ij}^1, \ldots, c_{ij}^\ell, \ldots, c_{ij}^p$ denote the vector of values assigned to each edge $\{i,j\} \in E$. The notation C is used to encompass all these vectors and our valued graph becomes $G = (V, E, C)$. Thus, when taking into account *multi-objective spanning tree* (*MOST*) problems, z is a function such that $z : \mathcal{T} \to \mathbb{R}_+^p$. This problem considers a vector of p objective functions, $z(T) = (z_1(T), \ldots, z_\ell(T), \ldots, z_p(T))$, each of which to be minimized, subject to the same constraint, i.e., $T \in \mathcal{T}$. The image of $\mathcal{T}$ according to all the objective functions, denoted by $Z = z(\mathcal{T}) \subset \mathbb{R}^p$, is the *feasible region in the objective space*.

A fundamental concept in multi-objective optimization is the *dominance* concept, which can be defined as follows (see [10, 36]).

Definition 2 (*Dominance*) Let $z', z'' \in \mathbb{R}^p$ denote two points in the objective space. Then, z' dominates z'', denoted by $z' \triangle z''$, if and only if $z' \leq z''$ and $z' \neq z''$ (i.e., $z'_\ell \leqslant z''_\ell$ for all $\ell = 1, \ldots, p$, and $z'_\ell < z''_\ell$ for $\ell \in \{1, \ldots, p\}$).

Consider a feasible point $\bar{z} \in Z$; this point is called a *non-dominated point* if and only if there is no other point $z \in Z$ such that $z \triangle \bar{z}$. Otherwise, $\bar{z}$ is a *dominated point*. Let N denote the set of all non-dominated points. A spanning tree $\bar{T} \in \mathcal{T}$ is said to be a *Pareto* or *efficient spanning tree* if and only if there does not exist another spanning tree $T \in \mathcal{T}$, such that $z(T) \triangle z(\bar{T})$; otherwise, it is called *inefficient*. Let P denote the set of all Pareto or efficient spanning trees.

Let $K^{\geq} = \{z \in \mathbb{R}^p : z_\ell \geqslant 0, \ell = 1, \ldots, p\}$ denote the *dominating cone* (non negative orthant). The *convex hull* of the set $N \oplus K^{\geq}$ (where $\oplus$ represents the set addition operation) is denoted by $Z^{\geq} = \text{conv}\ (N \oplus K^{\geq})$. The boundary of $Z^{\geq}$ is denoted by $\text{bd}\ (Z^{\geq})$, while its interior is denoted by $\text{int}\ (Z^{\geq})$.

Consider a non-dominated point $\bar{z} \in N$; this point is called a *supported non-dominated point* if and only if $\bar{z} \in \text{bd}\ (Z^{\geq})$. If $\bar{z}$ is a supported non-dominated point and it is also an extreme point of $Z^{\geq}$, $\bar{z}$ is said to be a supported *extreme non-dominated point*. A point $\bar{z} \in Z$ is said to be an *unsupported non-dominated point* if and only if $\bar{z} \in N$ and $\bar{z} \in \text{int}\ (Z^{\geq})$. The same concepts apply to the decision space. Finally, let *NS* denote the set of all supported non-dominated points and let *PS* the set of all supported efficient spanning trees.

2.3 The weighted-sum spanning tree problem

Our approach is based on weight set decomposition to define adjacency amongst spanning trees. A fundamental concept to that decomposition is the *weight set*, which is defined as follows:

Definition 3 (*Weight set*) The *weight set*, $W \subset \mathbb{R}^p$, can be stated as follows,

$$W = \left\{ w \in \mathbb{R}^p : \sum_{\ell=1}^{p} w_\ell = 1, \ \text{with} \ w_\ell > 0, \ \text{for}\ \ell = 1, \ldots, p \right\}.$$

Note that the dimension of the weight set, W, is $p-1$, denoted by $\dim(W) = p-1$, since $w_p = 1 - \sum_{\ell=1}^{p-1} w_\ell$.

Definition 4 (*Parametric problem*) The *parametric* or *weighted-sum problem* (*wMOST*), for some *parameter* or *weight vector* $w \in W$, can be defined as follows.

$$\min z^w(T) = \sum_{\ell=1}^{p} w_\ell z_\ell(T), \quad \text{for all} \quad T \in \mathcal{T}. \tag{1}$$

The optimal spanning trees (points) of this *w MOST* problem are supported efficient spanning trees (points) of the *MOST* problem (see [10, 15]).

Definition 5 (*Indifference region*) Let $\{z^1, \ldots, z^i, \ldots, z^r\}$, denote the set of non-dominated supported points. All the weighting vectors, $w \in W$, that lead to the same non-dominated supported point, $z^i \in Z^{\geq}$, for $i = 1, \ldots, r$, of the MOST problem, form the set W^i, which is referred to as the *indifference region* in the weight set associated with point z^i.

In the literature, some authors have proposed approaches to compute the set of extreme supported non-dominated points. Przybylski et al. [27] and Bökler et al.[5] compute a weight set decomposition, while Özpeynirci et al. [25] identify non-

dominated faces. Alves and Costa [3] proposed an algorithm to obtain all extreme supported non-dominated solutions adjacent to a given solution by exploring the weight space for multi-objective integer and mixed-integer linear programming problems. The algorithms are based on an exploration of the weight set taking advantage of geometrical properties of indifference regions. Moreover, several properties related to indifference regions are presented in Przybylski et al. [27].

Proposition 3 (Properties of an indifference region)

(a) *Extreme point: A non-dominated point,* z^i, *is an extreme point of* $Z^{\geq}$, *if and only if* W^i *is a polytope of dimension* $p-1$;
(b) *Adjacency in the objective space: Two non-dominated extreme points,* $z^i, z^j \in Z^{\geq}$, *are said to be* adjacent *if and only if* $W^i \cap W^j$ *is a polytope of dimension* $p-2$.
(c) *Weight set decomposition: Let* $\{z^1, \ldots, z^i, \ldots, z^r\}$ *denote the set of non-dominated extreme points of* $Z^{\geq}$, *and let* W^i *denote the indifference region associated with point* z^i, for $i = 1, \ldots, r$. *It holds that* $W = \bigcup_{i=1}^{r} W^i$.

In this article, we propose an algorithm to decompose the weight set and to enumerate the whole set of supported efficient trees from this decomposition. From Proposition 3(a) and (c) it can be inferred that the number of extreme supported points is equal to the number of polytopes of dimension $p-1$ (sub-divisions of the weight set). Thus, an interior point from each of these polytopes can be used to provide the weights of a weighted-sum MOST problem to obtain all extreme supported trees. Non-extreme supported trees can also be enumerated by finding the appropriate weights for weighted-sum problems. These weights correspond to interior points of intersections between indifference regions of dimension $p-1$. Interior points of regions corresponding to intersections of indifference regions of extreme solutions can be used to create weights for the weighed-sum MOST problem, where these non-extreme supported trees are optimal solutions. In Sect. 3, we provide a detailed description and a correctness proof of the algorithm.

2.4 Solution approaches to the MOST problem

Literature on the MOST problem is rather scarce. For the bi-objective problem, Ahuja et al. [2] developed a parametric programming approach (see Definition 4) to find the set of supported trees by using the concept of adjacent spanning trees (see Definition 1) in a network model (i.e., a neighborhood search technique through elementary adjacent moves). Their approach starts from a lexicographic optimal spanning tree for one of the objective functions and, by the adjacency relation, moves to the next supported spanning tree until the optimal spanning tree for the second objective is reached.

Two two-phase based methods have been proposed to find all non-dominated points in the objective space [29, 35]. In Ramos et al. [29], the first phase finds the set of all supported spanning trees by using a different technique from the one proposed in Ahuja et al. [2]. Their approach starts from two lexicographical optimal

spanning trees for both objectives and finds the remaining spanning trees by iteratively bisecting the objective space for each pair of points corresponding to the supported spanning trees that were previously found (see also [18]). The second phase finds the remaining efficient spanning trees with a branch-and-bound algorithm, which makes use of the information provided by a continuously updated set of efficient spanning trees, and a lower bound on each partial solution, to avoid branching.

Sourd and Spanjaard proposed a branch and bound, initialized with a set of extreme supported solutions, using neighborhood search after the first initialization, and by applying tighter bounds within the branch-and-bound algorithm [34]. Note that their approach only computes one efficient spanning tree for each non-dominated point.

The approach proposed in Steiner et al. [35] constructs the set of supported solutions as in Ramos et al. [29], but it makes use of *k*-best solutions based algorithm for the second phase, in order to search for the remaining efficient spanning trees. This algorithm solves a single-objective minimum spanning tree problem for each pair of consecutive supported spanning trees. In this new single-objective problem, the costs of the edges are obtained by weighting the costs of both objectives in relation to the costs of the consecutive pair of trees. The *k*-best algorithm is run to obtain the remaining spanning trees. Similar approaches have been proposed for obtaining an approximation of the non-dominated set. In Hamacher et al. [18], the first phase follows the same principle as in Ramos e-t al. [29]. The second phase is based on neighborhood search. Amongst each pair of consecutive spanning trees in the current set, ordered according to one of the objectives, the approach searches for a spanning tree that is not under a certain euclidean distance between the corresponding objective points from the pair. Andersen et al. [4] introduced a similar two-phase based technique, but considering further variants in the second phase.

Other approximation methods have been proposed, in particular, evolutionary based algorithms. Zhou et al. [39] proposed an evolutionary algorithm approach to find an approximation of the non-dominated set. This approach makes use of the Prufer encoding to represent a spanning tree. Knowles et al. [21] also proposed an evolutionary algorithm that makes use of three encoding techniques as well as evolutionary operators for each encoding.

Seipp introduced an approach to find the set of non-dominated extreme points of the tri-objective spanning tree by using a weighted space decomposition technique [33]. The weight set is divided into the maximal possible divisions for which all the points in such a subspace have the same MST in the objective space for the parametric spanning tree problem. Then, for each subspace, an exact algorithm for the MST, such as the Kruskal algorithm, can be applied to obtain the corresponding supported non-dominated point. The author was mainly interested in showing that the number of non-dominated extreme points is polynomially bounded with respect to the instance size for a fixed number of objectives. No implementation of the algorithm has been made, and no description on how to efficiently encode and store the candidate indifference regions was proposed.

Corley [9] proposed a generalization of the Prim's algorithm [26] to find the whole set of efficient spanning trees. However, the approach was shown to be incorrect in Hamacher and Ruhe [18]. More recently, Lacour proposed to integrate a ranking strategy in a branch-and-bound algorithm [23]. The ranking procedure of the algorithm starts by finding a subset of supported trees and improves that set with a neighbourhood search. Then, the efficient spanning trees of the problem are found by several runs of the ranking procedure with different ranking directions. This algorithm is integrated in a branch-and-bound scheme, where the ranking procedure runs at a branching node level of the search tree rather than for the whole problem. Pugliese et al. [28] proposed a dynamic programming approach that builds all the spanning trees in a Prim-like scheme. Each partial solution is a spanning tree of a subset of vertices and is extended by adding an edge connecting it to a vertex that does not belong to such a subset. A dominance relation is used to prune partial solutions that represent spanning trees of the same subset of vertices.

Our review of the existing methods suggests that several techniques have been applied to design efficient algorithm for solving the bi-objective spanning tree problem. Most of these algorithms make use of bounds for pruning the search space. We also noticed that, for problems with three or more objectives, the solution approaches are rather scarce and elementary; they make no use of advanced pruning techniques. For example, the dynamic programming approach of Pugliese e-t al. [28] could certainly benefit of using a good bound to prune candidates even further. For bi-objective problems, an efficient bound that is quite often used is the set of supported non-dominated points, see [14] for the bi-objective knapsack problem. However, when dealing with more than two objectives, it is not trivial to devise an algorithm that efficiently calculates such a set. For instance, Lacour devised a very elementary method, which computes only a subset of supported non-dominated points using parametric programming for a "well dispersed" subset of weight sets [23]. Seipp proposed to calculate the full set of extreme points by dividing the weight set in several sub-spaces and solving a single objective problem for each subspace [33]. The computation of all weight sets would require that generated sets would be efficiently stored while they are being computed. There can exist at most $m = n(n-1)/2$ edges for an instance of size n, which means that $m(m-1)/2$ different polylines (corresponding to edge swaps) may intersect the weight set and split them into different weight sets. This result bounds the number of different $p-2$ weight sets that can be decomposed (see Proposition 3). Such an algorithm seems to require efficient implementations with regards to memory usage and time performance. Nevertheless, the work in Seipp does not give any details on how to implement it [33]. Additionally, this algorithm would not take into account that optimal solutions for neighboring indifference regions have adjacency relations between them, and it can be infered from the edge swaps that generate the different regions. Instead, several independent runs of the single objective versions of the problem need to be made in order to identify those solutions. Finally, this algorithm only provides a single solution per each extreme supported non-dominated point, ignoring alternative solutions that can be of interest for a decision maker, and ignoring non-extreme supported efficient solutions.

In the following, we introduce an algorithm that is able to compute the whole set of supported non-dominated points and supported solutions, avoiding to run several times an MST algorithm.

3 An algorithm for generating all supported spanning trees

This section presents an algorithm to obtain the set of supported efficient spanning trees. It starts by introducing some additional theoretical concepts. Then, it presents a description of the pseudo-code of the algorithm. The proof of its correctness is also presented. Finally, an example on a tri-objective problem illustrates the main steps of the algorithm.

3.1 Some additional concepts

In the following, we introduce the concept of *candidate* spanning tree, which is important for the connectedness of the set of efficient spanning trees.

Definition 6 (*Candidate spanning tree*) Let T' denote a supported spanning tree and T'' one of its immediate neighbors. T'' is called a *candidate spanning tree* if and only if:

(a) T' and T'' does not dominate each other (the elementary move, or *swapping*, that allows to obtain T' from T'' is called an *eligible move*).
(b) $z(T')$ and $z(T'')$ are adjacent points in $Z^{\geq}$ (T'' is obtained after *filtering* the eligible moves that bound the indifference region of T'. In what follows we call them *candidate moves*).

There is a relation between the extreme points of $Z^{\geq}$ and the corresponding spanning trees. For each extreme point of $Z^{\geq}$ there is at least one corresponding spanning tree. Moreover, the adjacency relation between extreme points of $Z^{\geq}$ has a correspondence with the adjacency of spanning trees (see [17]). In fact, according to the notion of adjacency in Definition 1, the set of extreme supported spanning trees is connected [17] and polynomial bounded (see page 77 in Seipp [33]).

Proposition 4 (Connectedness) *The set of extreme supported spanning trees is* connected.

Proposition 5 (Cardinality) *The number of extreme supported non-dominated points of a MOST problem is at most* $2m^{2(p-1)}$.

In the following, we introduce the concept of separating hyperplane, which is useful to show that our algorithm calculates only and all the supported efficient spanning trees. Consider a swap $s^{T',T''} = \langle \{i,j\} \leftrightarrow \{k,\ell\} \rangle$, allowing a move from T' to T''. For a given weight vector w, let $c^w_{ij} = \sum_{q=1}^{p} c^q_{ij} w_q$ denote the cost of vertex $\{i,j\} \in T'$, and $c^w_{k\ell} = \sum_{q=1}^{p} c^q_{k\ell} w_q$ denote the cost of vertex $\{k,\ell\} \in T''$ in the wMOST problem. Let $H \in \mathbb{R}^p$ denote a subspace of dimension $p-1$. H is called an

hyperplane of $\mathbb{R}^p$ and separates $\mathbb{R}^p$ in two convex sets, called *half spaces*. The half spaces can be *closed half spaces* if each space contains H (denoted by $H^{\leq}$ and $H^{\geq}$), or can be *open half spaces* if they do not contain H (denoted by $H^{<}$ and $H^{>}$). Let $H_{ij,k\ell} = \{w \in \mathbb{R}^p : c_{ij}^w = c_{k\ell}^w\}$ denote an hyperplane that divides $\mathbb{R}^p$ into the closed half spaces, $H_{ij,k\ell}^{\leq} = \{w \in \mathbb{R}^p : c_{ij}^w \leqslant c_{k\ell}^w\}$ and $H_{ij,k\ell}^{\geq} = \{w \in \mathbb{R}^p : c_{ij}^w \geqslant c_{k\ell}^w\}$. Note that the set $H_{ij,k\ell} \cap W$ contains all the weight vectors w for which edges $\{i,j\}$ and $\{k,\ell\}$ weight the same for the parametric spanning tree problem. From this observation, the definition of a *separating hyperplane* can be devised as follows (see page 61 in [33]).

Proposition 6 (Separating hyperplane) *Consider a weight set $W' \subseteq W$, and an hyperplane $H_{ij,k\ell}$. The hyperplane $W_{ij,k\ell} = H_{ij,k\ell} \cap W'$ is called a* separating hyperplane *if and only if $dim(W_{ij,k\ell}) = p - 2$.*

3.2 The general algorithmic framework

The algorithm presented in this section generates the whole set of efficient supported spanning trees and corresponding supported non-dominated points of the MOST problem by applying local search techniques in a neighborhood that takes into account the definition of adjacency (see Definition 1). As input the algorithm requires the valued graph, $G = (V, E, C)$, and the weight set W. The main output is the above mentioned sets of supported efficient solutions and supported non-dominated points. In addition, the algorithm also outputs W split into all indifference regions $W^1, \ldots, W^i, \ldots, W^r$.

Algorithm 1 starts by finding a *lexicographic* efficient spanning tree T'. This is performed by ordering the edges of E in a lexicographic way, with respect to the values of each objective function, $z_1, \ldots, z_\ell, \ldots, z_p$. For calculating such a tree, we use the function, *lexicographic_MST*(G), which requires as input the graph G. At each step, the algorithm adds, to the current forest, the least lexicographically ordered edge.

In what follows, we shall consider two lists:

- $M(T')$ is a list containing all the filtered *eligible elementary moves* that can be performed from T'; function *swap_filter*(T') lists those moves.
- L is a list containing *extreme supported spanning trees*; function *extract(L)* extracts a candidate spanning tree T', and updates L;

The fundamental goal of the main loop of the algorithm is to calculate the indifference region associated with an extreme supported spanning tree, T', and to generate the supported spanning trees neighbors of T'. It works as follows: While the list L is not empty, a tree, T', is extracted from this list, through the use of the function *extract*(L). Then, the function *swap_filter*(T') is applied to generate the list of eligible elementary moves that can be applied to T'. This filtering is a performance improvement over the algorithm proposed in Seipp [33]. The swaps that are filtered are those that lead to infeasible, dominated and equivalent solutions.

These swaps do not contribute to the generation of the indifference region, as it will be proved in the following section.

The indifference region W' is calculated by using the function *build_indif*(T, $M(T'), W$). Note that all the spanning trees associated to two extreme points that bound this region, and all the spanning trees associated to eventual supported non-extreme points in that face, are optimal for the parametric problem for every point $w \in W'$. Note also that the optimal trees for the parametric spanning tree problem corresponding to points that belong to intersections of boundaries of W' are the union of the optimal trees of the parametric problem for every point in these boundaries. Thus, for each pair of boundaries of W' with a non-empty intersection, its intersection point needs to be calculated. For all boundaries with no intersections with any other boundary, an interior point is computed. All minimum spanning trees corresponding to parametric problems for weights based on weight set points calculated as described are computed. All those trees are added to the set of supported trees, *PS*, and their correspondent objective points are added to the set *NS*. L is updated with the inclusion of one of the trees corresponding to the newly calculated point, which is potentially a candidate extreme point. In order to improve CPU-time performance, and correctly halt the algorithm when all the points have been calculated, some additional lists are added to the algorithm: A list that contains all the supported extreme points already computed and a list containing points of the weight set that were already expanded. This is done to avoid recomputing weight set points and indifference regions.

In the following, we give some more details about the procedures and their time complexity:

- *extract*(L): This function extracts a tree T', from the list of candidate spanning trees, L, according to a heuristic selection rule (e.g., first-in-first-out, last-in-last-out, randomly). This function runs in $\mathcal{O}(1)$ and the order in which the trees are selected does not affect its performance (note that choosing a different selection rule may yield to a different performance in practice. However, there does not seem to exist an obvious reason to choose a different rule in our case).
- *swap_filter*(T'): This function is divided into two main phases: (i) The generation of eligible moves; and (ii) the filter of those moves by considering only the candidate moves.

  (i) In the first phase, the list of eligible moves that can be applied to T' is created. This step is performed in linear time with respect to the maximum number of moves for each edge of the tree (the total number of moves *per* tree is bounded above by $(n-1)(m-1)$).

  (ii) In the second phase, the filtering of moves occurs. A eligible move $s^{T',T'''}$ is removed from the list of candidate moves if, and only if, there exists another eligible move $s^{T',T''}$ in which $W_{T',T'''} \subset W_{T',T''}$. Then, each eligible move is compared against each other. Consequently, and as the number of eligible moves is bounded above by $(n-1)(m-1)$, the overall complexity of the filtering is $\mathcal{O}(n^2m^2)$.

- *build_indif*$(T', M(T'), W)$: This function calculates the weight set that represents the indifference region of the tree T'. This task is performed by considering initially the weight set W, and by reducing such set by generating the separating hyperplane associated to each candidate move. Considering t candidate moves, the number of operations of this function is bounded by $\mathcal{O}(t^2)$. Note that the maximum number of candidate moves is bounded above by the minimum between the number of extreme spanning trees ($2m^{2(p-1)}$) and the number of possible eligible moves $(n-1)(m-1)$. Thus, the complexity of this function is bounded above by $\mathcal{O}(\min\{n^2m^2, m^{4(p-1)}\})$. Each indifference region is represented by the vertices of the boundary of each region.
- *build_ST*$(T', s^{T',T''})$: This function generates a candidate spanning tree, T'', by copying all the edges of the supported extreme spanning tree, T', but one, and swapping the edges according to the move $s^{T',T''}$. The complexity of copying a tree is linear with respect to the number of edges of each tree $(n-1)$. The swapping operation can be performed either when copying the original tree, or in a separated step. In the first case, the swap is done in constant time, while in the second case it runs linearly with respect to the number of edges. The overall complexity is thus $\mathcal{O}(n)$, independently on the strategy to select the swapping of edges.
- *computed_weight_points*(W^i): This function receives the weight set of dimension $p-2$ calculated in *build_indif*, W', and returns points of this weight set, in such a manner that all supported points of the problem can be computed. Each $p-2$-dimensional weight set is delimited by $p-3$-dimensional weight sets, and, in general, each $p-k$-dimensional weight set is delimited by $p-(k+1)$-dimensional weight sets, for all $k<p$. Thus, W^i is delimited by weight sets of dimension k where $0 \leq k \leq p-2$. For each of this delimiting weight sets, at least one interior point is added and returned by this function.

  (a) For each pair of 0-dimensional delimiting point of W^i that does not belong to the weight set, it is checked if they belong to any 1-dimensional delimiting weight set of W^i and it belongs to the weight set. If affirmative, an interior point of this 1-dimensional delimiting weight set is returned. If not, the 1-dimensional weight is added to a new temporary set to be used as it is described in the following:
  (b) Procedure (a) is recursively applied for each k-dimensional delimiting sets, for $1 \geq k \geq p-2$, stored in the temporary set in the previous iteration, until no $k+1$-dimensional weight sets are added, or $k = p+2$. All interior points found in such a way are returned by the function.

- *all_MST*$(G = (V, E, wC))$: This function receives the graph, with vertices, and edges, and the weights for the costs of each edge. It returns all the minimum spanning trees of that problem. After computing this function, the corresponding indifference region can be discarded. This is performed for efficiency reasons,

since although the number of indifference regions is polynomially bounded, it may require a large amount of memory.

- *candidate*($PS^{i,k}$): This function receives a set of supported spanning trees and returns the list of candidate supported extreme trees that has not been computed so far.

Algorithm 1 Listing all the supported spanning trees for the MOST problem.

Require: $G = (V, E, C), W$;
Ensure: $NS, PS, \{W^1, \ldots, W^i, \ldots, W^r\}$;
1: $T' \leftarrow lexicographic\_MST(G)$;
2: $L \leftarrow \{T'\}$;
3: $NS \leftarrow \{z(T')\}$;
4: $PS \leftarrow \{T'\}$;
5: $i \leftarrow 1$;
6: **while** $(|L| > 0)$ **do**
7: $\quad T' \leftarrow extract(L)$;
8: $\quad M(T') \leftarrow swap\_filter(T')$;
9: $\quad W' \leftarrow build\_indif(T, M(T'), W)$;
10: $\quad W^i \leftarrow W'$;
11: $\quad w^\star \leftarrow computed\_weight\_points(W^i)$;
12: $\quad$ **for** $w \in w^\star$ **do**
13: $\quad\quad PS^i \leftarrow all\_MST(G = (V, E, wC))$;
14: $\quad\quad PS \leftarrow PS \cup PS^i$;
15: $\quad\quad NS \leftarrow NS \cup \{z(PS^i)\}$;
16: $\quad\quad L \leftarrow L \cup candidate(PS^i)$;
17: $\quad\quad i \leftarrow i + |PS^i|$;
18: **return** $NS, PS, \{W^1, \ldots, W^i, \ldots, W^r\}$;

3.3 Correctness of the algorithm

Our algorithm is based on the solution of parametric problems and a neighborhood search technique. It makes use of the concept of indifference region, and starts by calculating a lexicographically MST, which is an extreme supported spanning tree. The algorithm solves the problem of determining all the supported efficient spanning trees.

Theorem 1 (Correctness of the algorithm) *Algorithm 1 outputs only all supported spanning trees for MOST problem.*

Proof The algorithm correctness consists of proving the following two statements.

1. If T' is found by the algorithm at step 1 or 14, then T' is an efficient supported spanning tree (soundness).
2. The algorithm finds all the supported efficient spanning trees (completeness), that is, at step 18, *NS* contains only and all supported efficient spanning trees.

Soundness is proved by construction, while completeness is done by contradiction. In case 1, we claim that, if T' is found by the algorithm, then T' is an efficient supported spanning tree. There are two possible cases:

(a) Either T' is an extreme supported efficient spanning trees, or;
(b) T' is a non-extreme supported efficient spanning tree.

In Case (a), T' has an indifference region of dimension $p-1$ (Proposition 3a), while in the Case (b), T' has an indifference region of at most dimension $p-2$ (Proposition 3b).

The algorithm works by building indifference regions adjacent to already calculated indifference regions. This is performed in such a manner that the whole indifference region W is created. Let $\{z^1, \ldots, z^i, \ldots, z^r\}$ denote the set of non-dominated extreme points, and W^i denote the indifference region associated to trees with objective value z^i, for all $i = 1, \ldots, r$. From Proposition 3(c) we know that

$$\bigcup_{i=1}^{r} W^i = W.$$

The algorithm stops when W is entirely built, and there is no supported efficient solution left to be explored, that is, list L is empty (Line 6, Algorithm 1). As described in Sect. 3.2, the edges of E are ordered in a lexicographic order with respect to the values of each objective function, $z_1(T), \ldots, z_\ell(T), \ldots, z_p(T)$, for all $T \in \mathcal{T}$. The first supported efficient tree T' is obtained by an application of the Kruskal algorithm, with edges ordered in a lexicographical order with respect to one of the objectives. It should be noted that at this step, only a single lexicographic minimum spanning tree is calculated. The corresponding non-dominated point in the objective space and the solution are added to the list of supported points/solutions (Line 3 and 4, Algorithm 1). If there are multiple optimal extreme supported spanning trees for this minimum lexicographic extreme non-dominated point, they will be computed after computing the indifference region for T'. (Lines 7–9, Algorithm 1) Thus, the $p-1$ dimensional indifference region, W', associated with T' is built by the algorithm.

The indifference region of T', W', is delimited by a set of indifference regions that are single points of the weight set, which are the intersection of W' with other regions. From Proposition 3b, we know that the parametric problem with weight sets of each of these indifference regions have at least one alternative optimal spanning tree. Thus, T' has at least the same number of adjacent spanning trees as the number of adjacent indifference regions W'. These adjacent spanning trees are extreme spanning tree solutions (Case a)).

Non-extreme supported efficient trees (Case b)) are optimal in points of the weight set that do not belong exclusively to a $p-1$ dimensional weight region (otherwise they would be extreme supported trees, according to Proposition 3a). Thus, these trees are optimal in points of $(p-i)$ dimensional indifference regions (for $p \geqslant i \geqslant 2$). From W' we can calculate the points to find such trees. These trees are calculated by finding the optimal solutions corresponding to the points extracted from the delimiting indifference regions computed by function *computed_weight_set* (where the adjacent spanning trees of T' are also alternative optimal).

The next step of the algorithm is to pick a vector of weights from each of the indifference regions delimiting W', where multiple optimal supported spanning trees can be found (Line 12, Algorithm 1). For each selected vector of weights, *all_MST*

algorithm is applied for the corresponding parametric spanning tree problem (Line 13, Algorithm 1). The union of the outcome of the computation of all runs of the *all_MST* procedure leads to a subset of supported efficient spanning trees (Lines 15–16, Algorithm 1). From this subset, the trees that have already been computed are discarded, whereas the remaining are added to the list of computed spanning trees. Note that at that point it is not possible to know if the new supported efficient spanning trees are extreme (Case *a*)) or not (Case *b*)). However, there is the guarantee that at least one of the new supported efficient solutions is extreme. The only exception is the case where there exists only one indifference region in the whole weight set. Therefore, the new supported efficient spanning trees are added to the list. (Line 15, Algorithm 1)

The algorithm is applied to all the efficient spanning trees in the candidate set (Line 6, Algorithm 1). The efficient spanning trees that generate $p-1$ dimensional indifference regions are extreme spanning trees (Case *a*)), and are used for computing new supported efficient spanning trees. The efficient spanning trees that generate indifference regions with at most $p-2$ dimensions, are non-extreme (Case *b*)) and this indifference region will not be used to further computations, as they were previously computed.

Note that T' can never be a non-supported spanning tree, otherwise it would have an empty indifference region, which is not the case for any spanning tree solution found by the algorithm, as either it is a lexicographic minimum (which has a corresponding $p-1$ dimensional indifference region), or it is found as an alternative optimal solution for a parametric problem of a given weight set, thus having a non-empty indifference region. In other words, by construction, it is clear that the set *NS* in Line 18 cannot contain inefficient or unsupported efficient spanning trees, since it makes use of the parametric problem. Thus, the output is always a supported efficient spanning tree.

We now prove 2). Assume that T' is a supported spanning tree that was not found by the algorithm. The only reason for this to happen is because the algorithm was not able to find the indifference region to which T' belongs to. However, since the indifference regions are connected, and since function *All_MST* finds all supported efficient spanning trees, it is not possible to have missed that indifference region. □

4 An illustrative example

This section introduces a small illustrative example with three objective functions. In the first subsection, we report all the MSTs for a parametric problem as well as the adjacency graph. Then, all the non-dominated points and corresponding efficient spanning trees are listed. Finally, we illustrate the main steps of the algorithm in this particular case.

4.1 A complete graph and the set of all optimal spanning trees

Figure 1 presents an example of a complete graph with 7 vertices and 21 edges. For this graph, the total number of spanning trees, according to the Cayley formula, is $n^{n-2} = 7^5 = 16807$.

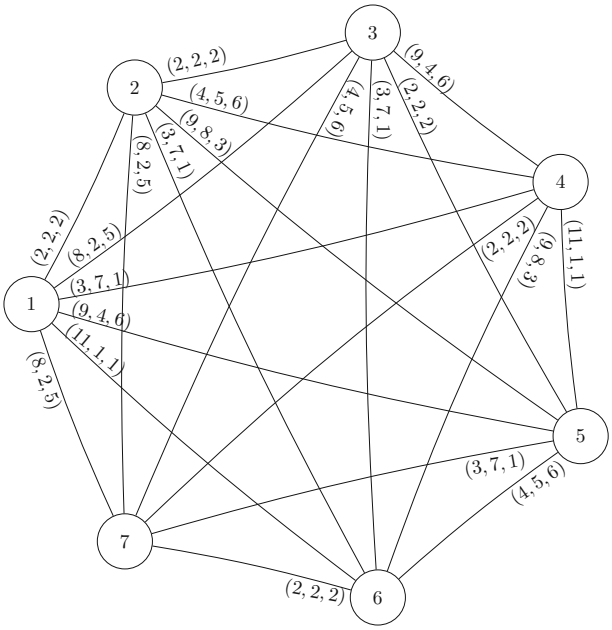

Fig. 1 Complete graph used to illustrate the algorithmic framework

Consider the parametric problem with weight $\lambda = (6/60, 4/60, 50/60)$. There is a total of 12 MSTs with $z^{\lambda}(T^*) = 52/5$ (see Table 1). Figure 2 shows the adjacency graph of these MSTs. This is in accordance with the result of Proposition 2.

Table 1 MSTs for the parametric problem

Set $\mathcal{T}^*$, for $\lambda_1 = 6/60$, $\lambda_2 = 4/60$, and $\lambda_3 = 50/60$
$T^{*,1} = \{\{1,2\},\{1,4\},\{2,6\},\{3,5\},\{3,6\},\{5,7\}\}$
$T^{*,2} = \{\{1,4\},\{2,6\},\{3,5\},\{3,6\},\{4,7\},\{5,7\}\}$
$T^{*,3} = \{\{1,4\},\{1,6\},\{2,6\},\{3,5\},\{3,6\},\{5,7\}\}$
$T^{*,4} = \{\{1,4\},\{2,6\},\{3,5\},\{3,6\},\{4,5\},\{5,7\}\}$
$T^{*,5} = \{\{1,2\},\{1,4\},\{2,6\},\{3,6\},\{4,7\},\{5,7\}\}$
$T^{*,6} = \{\{1,2\},\{1,4\},\{2,6\},\{3,6\},\{5,7\},\{6,7\}\}$
$T^{*,7} = \{\{1,2\},\{1,4\},\{2,6\},\{4,5\},\{4,7\},\{5,7\}\}$
$T^{*,8} = \{\{1,4\},\{2,6\},\{3,6\},\{4,7\},\{5,7\},\{6,7\}\}$
$T^{*,9} = \{\{1,4\},\{1,6\},\{2,6\},\{3,6\},\{4,7\},\{5,7\}\}$
$T^{*,10} = \{\{1,4\},\{1,6\},\{2,6\},\{3,6\},\{5,7\},\{6,7\}\}$
$T^{*,11} = \{\{1,4\},\{1,6\},\{2,6\},\{3,6\},\{4,5\},\{5,7\}\}$
$T^{*,12} = \{\{1,4\},\{2,6\},\{3,6\},\{4,5\},\{5,7\},\{6,7\}\}$

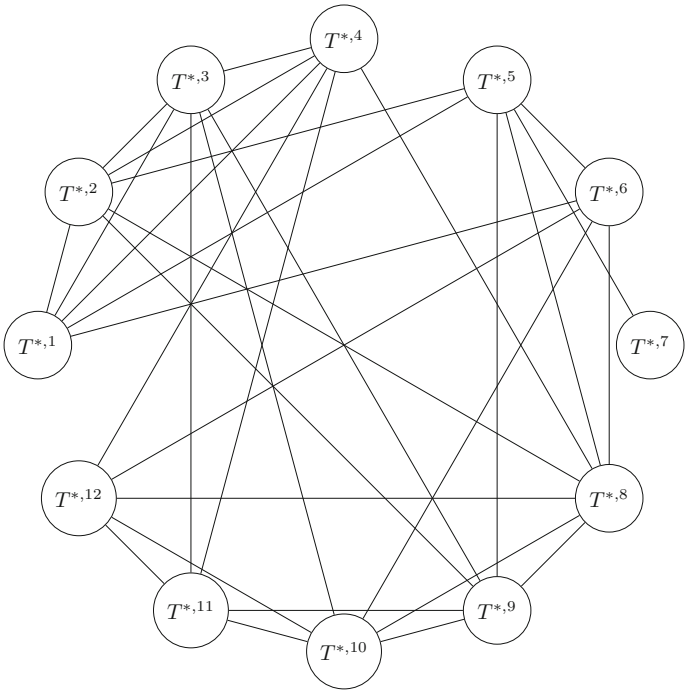

Fig. 2 Adjacency graph of the optimal solutions of the parametric problem for $\lambda_1 = 6/60$, $\lambda_2 = 4/60$, and $\lambda_3 = 50/60$

4.2 All efficient spanning trees

This example has 16 non-dominated points:

$$N = \{z^1 = (13, 17, 11), z^2 = (14, 15, 16), z^3 = (18, 12, 15), z^4 = (21, 11, 11),$$
$$z^5 = (30, 10, 10), z^6 = (16, 32, 8), z^7 = (34, 30, 6), z^8 = (14, 22, 10),$$
$$z^9 = (15, 27, 9), z^{10} = (31, 15, 9), z^{11} = (32, 20, 8), z^{12} = (33, 25, 7),$$
$$z^{13} = (25, 31, 7), z^{14} = (22, 16, 10), z^{15} = (23, 21, 9), z^{16} = (24, 26, 8)\}$$

All 16 points are supported, of which z^1, z^2, z^3, z^4, z^5, z^6 and z^7 are extreme. Figure 3 shows a three-dimensional representation of the set of non-dominated points. Points in red correspond to supported extreme, points in green color correspond to supported non-extreme points that are on the edges of the convex hull defined by non-dominated points, and points in black are supported non-extreme points that are not in those edges. The non-dominated facets are also represented. "Appendix A" presents all the efficient spanning trees associated with all non-dominated points for this particular instance.

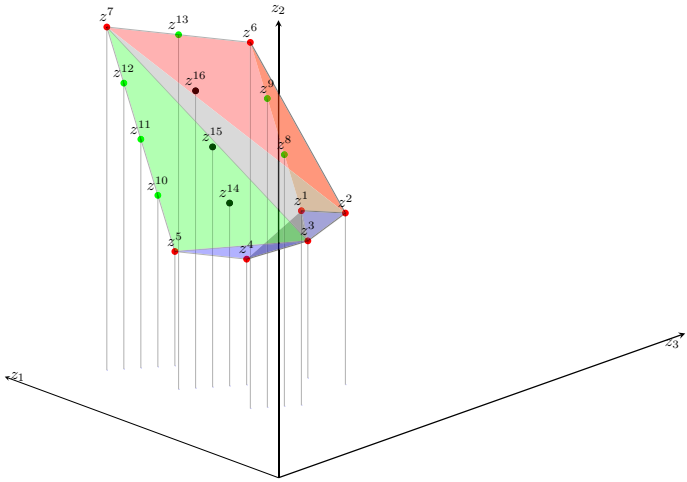

Fig. 3 Non-dominated points of the problem

4.3 Computing all the supported spanning trees

In the following, we ilustrate the main steps of Algorithm 1 for the illustrative example. First, a lexicographic spanning tree T' is calculated (Line 1), where $z(T') = (13, 17, 11)$. This tree is added to list L (Line 2) and, since T' is an extreme supported tree, sets NS and PS are updated (Lines 3 and 4). T' is the first tree, thus, from hereafter, it is referred to as T^1. Variable i is updated (Line 5).

Then, the main loop starts (Line 6). T^1, the only tree in L, is extracted from L (now temporarily empty) to T' (Line 7). Method *swap_filter* is applied to T' to extract the elementary moves that lead to immediate neighbors of T'. Table 2 shows the list of elementary moves that lead to trees that are not dominated by T^1 (Line 8). This list of elementary moves is filtered out in order to contain the minimum amount of edge swaps that allow the generation of the indifference region W^1 associated to T^1 (Line 9).

Note that there are 24 edge swaps that lead to spanning trees that are not dominated by T^1, but they only generate 6 distinct separating hyperplanes. Figure 4 displays the 6 distinct hyperplanes. The gray filled area represents the weight set. This weight set corresponds to the region $\lambda_1 > 0, \lambda_2 > 0, \lambda_1 + \lambda_2 < 1$, therefore, it is delimited by $\lambda_1 = 0$, $\lambda_2 = 0$ and $\lambda_1 = 1 - \lambda_2$, but they are not included in the set. Note that T^1 is a lexicographical optimal solution considering that λ_1 refers to the first objective. First, only one swap from each group of distinct separating hyperplanes is kept by the filter function. Thereafter, pairwise comparisons amongst the remaining swaps are performed. The goal of those comparisons is to find if any other edge swap can be removed before calculating the indifference region W^1. A swap s is filtered if, when calculating W^1, there is at least one other swap that would reduce the region in such a way that the separating hyperplane generated by s would not intersect the reduced region. In this example, the separating hyperplane

Table 2 Elementary moves associated to T^1

Edge out	Edge in	Separating hyperplane	$\lambda_1 = 0$	$\lambda_2 = 0$	$\lambda_1 = \lambda_2$
$\{1,2\}$	$\{2, 6\}$	$\lambda_2 = -\frac{1}{3}\lambda_1 + \frac{1}{6}$	$\lambda_2 = \frac{1}{6}$	$\lambda_1 = \frac{1}{2}$	$\lambda_1 > 1$
$\{1,2\}$	$\{3, 6\}$	$\lambda_2 = -\frac{1}{3}\lambda_1 + \frac{1}{6}$	$\lambda_2 = \frac{1}{6}$	$\lambda_1 = \frac{1}{2}$	$\lambda_1 > 1$
$\{1,2\}$	$\{4, 5\}$	$\lambda_1 = 0.1$	–	$\lambda_1 = 0.1$	$\lambda_1 = 0.1$
$\{1,2\}$	$\{5, 7\}$	$\lambda_2 = -\frac{1}{3}\lambda_1 + \frac{1}{6}$	$\lambda_2 = \frac{1}{6}$	$\lambda_1 = \frac{1}{2}$	$\lambda_1 > 1$
$\{2,3\}$	$\{3, 6\}$	$\lambda_2 = -\frac{1}{3}\lambda_1 + \frac{1}{6}$	$\lambda_2 = \frac{1}{6}$	$\lambda_1 = \frac{1}{2}$	$\lambda_1 > 1$
$\{2,3\}$	$\{4, 5\}$	$\lambda_1 = 0.1$	–	$\lambda_1 = 0.1$	$\lambda_1 = 0.1$
$\{2,3\}$	$\{5, 7\}$	$\lambda_2 = -\frac{1}{3}\lambda_1 + \frac{1}{6}$	$\lambda_2 = \frac{1}{6}$	$\lambda_1 = \frac{1}{2}$	$\lambda_1 > 1$
$\{1,4\}$	$\{1, 6\}$	$\lambda_2 = \frac{4}{3}\lambda_1$	$\lambda_2 = 0$	$\lambda_1 = 0$	$\lambda_1 = \frac{3}{7}$
$\{1,4\}$	$\{1, 7\}$	$\lambda_2 = \frac{1}{9}\lambda_1 + \frac{4}{9}$	$\lambda_2 = \frac{4}{9}$	$\lambda_1 < 0$	$\lambda_1 = 0.5$
$\{1,4\}$	$\{2, 4\}$	$\lambda_2 = -\frac{4}{7}\lambda_1 + \frac{5}{7}$	$\lambda_2 = \frac{5}{7}$	$\lambda_1 > 1$	$\lambda_1 = \frac{2}{3}$
$\{1,4\}$	$\{2, 7\}$	$\lambda_2 = \frac{1}{9}\lambda_1 + \frac{4}{9}$	$\lambda_2 = \frac{4}{9}$	$\lambda_1 < 0$	$\lambda_1 = 0.5$
$\{1,4\}$	$\{3, 4\}$	$\lambda_2 = \frac{1}{8}\lambda_1 + \frac{5}{8}$	$\lambda_2 = \frac{5}{8}$	$\lambda_1 < 0$	$\lambda_1 = \frac{1}{3}$
$\{1,4\}$	$\{3, 7\}$	$\lambda_2 = -\frac{4}{7}\lambda_1 + \frac{5}{7}$	$\lambda_2 = \frac{5}{7}$	$\lambda_1 > 1$	$\lambda_1 = \frac{2}{3}$
$\{1,4\}$	$\{4, 5\}$	$\lambda_2 = \frac{4}{3}\lambda_1$	$\lambda_2 = 0$	$\lambda_1 = 0$	$\lambda_1 = \frac{3}{7}$
$\{1,4\}$	$\{5, 6\}$	$\lambda_2 = -\frac{4}{7}\lambda_1 + \frac{5}{7}$	$\lambda_2 = \frac{5}{7}$	$\lambda_1 > 1$	$\lambda_1 = \frac{2}{3}$
$\{3,5\}$	$\{4, 5\}$	$\lambda_1 = 0.1$	–	$\lambda_1 = 0.1$	$\lambda_1 = 0.1$
$\{3,5\}$	$\{5, 7\}$	$\lambda_2 = -\frac{1}{3}\lambda_1 + \frac{1}{6}$	$\lambda_2 = \frac{1}{6}$	$\lambda_1 = \frac{1}{2}$	$\lambda_1 > 1$
$\{4,7\}$	$\{1, 6\}$	$\lambda_1 = 0.1$	–	$\lambda_1 = 0.1$	$\lambda_1 = 0.1$
$\{4,7\}$	$\{2, 6\}$	$\lambda_2 = -\frac{1}{3}\lambda_1 + \frac{1}{6}$	$\lambda_2 = \frac{1}{6}$	$\lambda_1 = \frac{1}{2}$	$\lambda_1 > 1$
$\{4,7\}$	$\{3, 6\}$	$\lambda_2 = -\frac{1}{3}\lambda_1 + \frac{1}{6}$	$\lambda_2 = \frac{1}{6}$	$\lambda_1 = \frac{1}{2}$	$\lambda_1 > 1$
$\{4,7\}$	$\{5, 7\}$	$\lambda_2 = -\frac{1}{3}\lambda_1 + \frac{1}{6}$	$\lambda_2 = \frac{1}{6}$	$\lambda_1 = \frac{1}{2}$	$\lambda_1 > 1$
$\{6,7\}$	$\{1, 6\}$	$\lambda_1 = 0.1$	–	$\lambda_1 = 0.1$	$\lambda_1 = 0.1$
$\{6,7\}$	$\{2, 6\}$	$\lambda_2 = -\frac{1}{3}\lambda_1 + \frac{1}{6}$	$\lambda_2 = \frac{1}{6}$	$\lambda_1 = \frac{1}{2}$	$\lambda_1 > 1$
$\{6,7\}$	$\{3, 6\}$	$\lambda_2 = -\frac{1}{3}\lambda_1 + \frac{1}{6}$	$\lambda_2 = \frac{1}{6}$	$\lambda_1 = \frac{1}{2}$	$\lambda_1 > 1$

represented by the straight line in gray in Fig. 4 ($\lambda_2 = \frac{1}{8}\lambda_1 + \frac{5}{8}$) is discarded, because it would not intersect a reduced region after computing the separating hyperplane in orange ($\lambda_2 = \frac{1}{9}\lambda_1 + \frac{4}{9}$).

Table 3 shows the edge swaps that return from function *swap_filter* (Line 8) and that are used as input for function *build_indif* (Line 9), to generate the indifference region W^1. The indifference region is intersecting the weight set region with each filtered separating hyperplane, and reducing the weight set at each step. The indifference region W^1 is shown in Fig. 5.

After calculating the indifference region, the $p-2$ dimensional facets that bound W^1 are obtained. These are the facets in which the alternative supported spanning trees are found, and from which the next extreme solutions can be calculated. In this

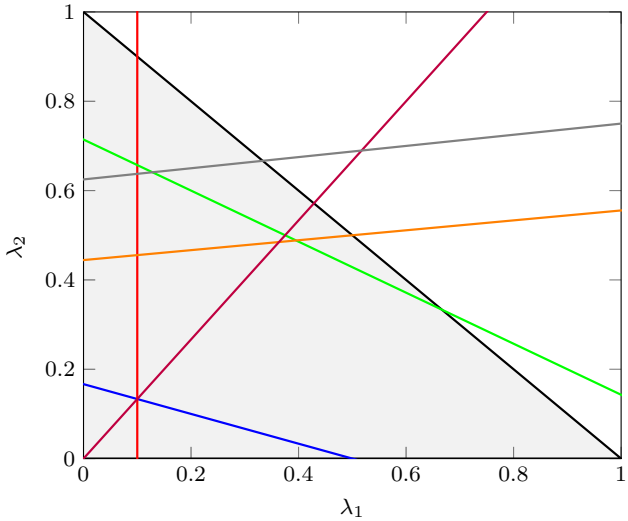

Fig. 4 Distinct separating hyperplanes generated by eligible swaps that lead to spanning trees that are not dominated by T^1

Table 3 Elementary moves associated to T^1 after the filtering

Edge out	Edge in	Separating hyperplane	$\lambda_1 = 0$	$\lambda_2 = 0$	$\lambda_1 = \lambda_2$
$\{1,4\}$	$\{2,7\}$	$\lambda_2 = \frac{1}{9}\lambda_1 + \frac{4}{9}$	$\lambda_2 = \frac{4}{9}$	$\lambda_1 < 0$	$\lambda_1 = 0.5$
$\{1,4\}$	$\{4,5\}$	$\lambda_2 = \frac{4}{3}\lambda_1$	$\lambda_2 = 0$	$\lambda_1 = 0$	$\lambda_1 = \frac{3}{7}$
$\{1,4\}$	$\{5,6\}$	$\lambda_2 = -\frac{4}{7}\lambda_1 + \frac{5}{7}$	$\lambda_2 = \frac{5}{7}$	$\lambda_1 > 1$	$\lambda_1 = \frac{2}{3}$
$\{6,7\}$	$\{1,6\}$	$\lambda_1 = 0.1$	–	$\lambda_1 = 0.1$	$\lambda_1 = 0.1$
$\{6,7\}$	$\{3,6\}$	$\lambda_2 = -\frac{1}{3}\lambda_1 + \frac{1}{6}$	$\lambda_2 = \frac{1}{6}$	$\lambda_1 = \frac{1}{2}$	$\lambda_1 > 1$

example, there are four $p-2$ facets that are used to calculate the alternative and upcoming spanning trees. This calculation is represented in Algorithm 1 in the inner *for* loop, and is explained as follows.

Each $p-2$ dimensional facet of W^1 is explored by computing weight points w^*, where $w^* \in W^1$ (Line 11) and for all facets $W^{1\prime} \in W^1$, at least one point $w' \in w^*$ belongs to $W^{1\prime}$. The algorithm retrieves all the spanning trees that are minimal for the parametric spanning tree problem with weights w (Line 13), for each $w \in w^*$. The trees and points found in this step are added to lists *PS* (Line 14) and *NS* (Line 15), respectively. Then, a spanning tree with objective function different than $z(T^1)$ per each facet is calculated and added to L (Line 16). In our numerical example a large number of alternative trees is calculated in this step.

Table 4 shows, in a simplified form, the trees calculated within these steps. A and B represent the points of the two 0-dimensional weight sets that bound each of the 1-

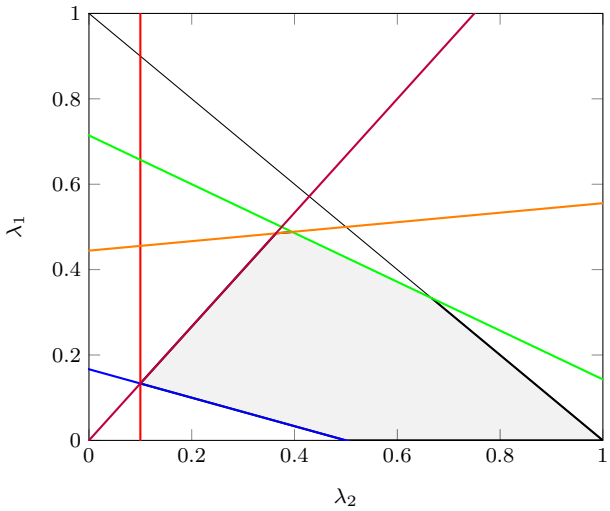

Fig. 5 Separating hyperplanes after filtering, and indifference region, W^1, associated to tree T^1

Table 4 Simplified steps of the inner *for* loop in Algorithm 1 for the numerical example

Facet	A	B	w	Alt. trees	New trees
$W^{1,1}$	(0.5, 0)	(0.1, 0.133)	(0.3, 0.067)	46	45
$W^{1,2}$	(0.1, 0.133)	(0.364, 0.485)	(0.232, 0.309)	6	2
$W^{1,3}$	(0.364, 0.485)	(0.395, 0.488)	(0.379, 0.487)	6	2
$W^{1,4}$	(0.395, 0.488)	(0.667, 0.333)	(0.531, 0.411)	7	3

dimensional weight set $W^{1\prime}$, while w represents an interior point of the 1-dimensional weight set. For a 3-objective problem, no more weight points than A, B and w needs to be taken into account as potential points to find supported non-dominated points of the problem. Some considerations about how to chose between A, B and w are provided below. Next column shows the number of alternative spanning trees that are optimal for that facet. The number of new trees corresponds to the number of supported trees that is first discovered in this facet. Note that in the first facet only one out of 46 trees is not new; such tree is T^1. After this procedure, 52 new supported spanning trees are found.

Note that it is not necessary to use A, B and w to find all supported trees. In fact, A and B belong to the 0-dimensional weight set, but they are also points that belong to the 1-dimensional point facet that we are computing. However, in some cases, A and B might fall outside the full weight set (e.g. $A = (0.5, 0)$, and $B = (1, 0)$), and, for these cases, the weight point w needs to be used to compute alternative supported points. Let us look at $W^{1,1}$. Point A is excluded as it lies outside the set of weights (it is in $\lambda_2 = 0$), thus all supported trees found in point w will also be found in B (but

not necessarily vice-versa). In case of $W^{1,2}$, where both A and B are inside the weight set, all the trees found in w will be present in A and in B, but eventually some trees in A will not be present in B and w, and some trees in B will not be present in A and w. This property also holds for problems with more than 3 objectives.

Table 5 displays the non-dominated points that are found in each facet, and one of the trees corresponding to each non-dominated point. Note that the last point presented in the table per each facet corresponds to the next extreme point. Therefore, trees T^4, T^5, T^6, and T^7 are added to list L.

In the next step of the inner *while* cycle, 4 trees are found in list $L = \{T^4, T^5, T^6, T^7\}$. The first one is extracted to T' and the process is repeated until L is empty. Note that, in the following steps, extreme trees corresponding to points calculated in a previous step are not included to L. This step is important to avoid generating cycles and calculating the same indifference region multiple times.

5 Computational experiments

This section presents an experimental analysis of our approach on a wide range of MOST instances. Algorithm 1 was implemented for MOST instances with 3 objective functions. The code was written in C++ programming language and compiled using compiler g++ with the optimization flag "–O3". The experiments were performed in an Intel (R) Core I7 4770 3.4 GHz with 32 GB Ram at 1600 MHz, with operating system Ubuntu version 16.04 LTS Xenial Xerus.

5.1 The design of the experiments

The majority of the experimental studies for this problem uses complete graphs [4, 29, 34], grid graphs [4, 34, 35], graphs with varying density [28, 35] and specifically modified hard graphs [34]. In most cases, the weight costs are generated

Table 5 Non-dominated points that are found in each facet

Facet	Supported point	Tree example
$W^{1,1}$	$\{13, 17, 11\}$	$T^1 = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{4,7\},\{6,7\}\}$
	$\{14, 22, 10\}$	$T^2 = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{4,7\},\{3,6\}\}$
	$\{15, 27, 9\}$	$T^3 = \{\{1,2\},\{3,6\},\{3,5\},\{4,7\},\{5,7\},\{1,4\}\}$
	$\{16, 32, 8\}$	$T^4 = \{\{1,2\},\{3,6\},\{3,5\},\{2,6\},\{5,7\},\{1,4\}\}$
$W^{1,2}$	$\{13, 17, 11\}$	$T^1 = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{4,7\},\{6,7\}\}$
	$\{21, 11, 11\}$	$T^5 = \{\{1,2\},\{2,3\},\{3,5\},\{4,7\},\{6,7\},\{4,5\}\}$
$W^{1,3}$	$\{13, 17, 11\}$	$T^1 = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{4,7\},\{6,7\}\}$
	$\{18, 12, 15\}$	$T^6 = \{\{1,2\},\{2,3\},\{3,5\},\{4,7\},\{6,7\},\{2,7\}\}$
$W^{1,4}$	$\{13, 17, 11\}$	$T^1 = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{4,7\},\{6,7\}\}$
	$\{14, 15, 16\}$	$T^7 = \{\{1,2\},\{2,3\},\{3,5\},\{4,7\},\{6,7\},\{5,6\}\}$

randomly, although, in some studies, correlated instances are also considered [4, 35]. Most graphs used on these studies have a relatively small amount of vertices (up to 50); only Sourd and Spanjaard in [34] report results on instances up to 500 vertices, and Steiner in [35] with uncorrelated grid instances up to 121 vertices. In Pugliese et al. [28], they report results on MOST instances with more than two objectives, with 10–20 vertices and, at most, 170 edges. Additionally, they have considered instances with 50–100 vertices but with a very low density of edges, from 62 to 125.

For our experimental analysis, three main types of graphs were considered:

1. Density graphs: graphs in which an edge between a pair of vertices is generated with a probability d ($d = 0.1, 0.4$, and 0.7).
2. Complete graphs: graphs with one edge connecting all pairs of vertices; note that a complete graph is a density graph with $d = 1.0$.
3. Grid: Vertices displayed in a $n \times n$ square, where each vertex is connected to the left, right, up and down neighbor vertices, except for the vertices in the limits of the grid.

The edge weight costs were randomly generated using an uniform distribution in the interval [1, 1000] for every objective value. Tests were performed for complete graphs from 10 to 60 vertices, density graphs from 10 to 110 vertices and grid graphs from 25 to 256 vertices. Additionally, a set of instances was generated with conflicting objectives in order to simulate a more realistic scenario. For those conflicting instances, the costs of the two first objectives, c^1_{ij}, c^2_{ij}, were generated similarly to non-correlated instances. The costs in the third objective were computed as follows: $c^3_{ij} = -\alpha(1 - c^1_{ij}) + (1 + \alpha)\beta$, where $\alpha \in [-1, 0]$ is the correlation parameter and β is a randomly generated value using an uniform distribution in the interval [1, 1000]. We consider $\alpha = -0.4$ (correlated instances) and $\alpha = -0.8$ (highly correlated instances) for our experiments. For each combination of graph type, density (whenever applicable), type of weight cost relation (random, conflicting or highly conflicting), and size, we generated 30 different test cases. Tables 6 and 7 characterize the density and grid graphs, respectively, in terms of the number of vertices.

Table 6 Average number of vertices and density

d	n										
	10	20	30	40	50	60	70	80	90	100	110
0.1	10.8	28.0	51.1	84.5	128.3	181.4	245.3	316.9	406.5	495.2	–
0.4	18.6	75.5	174.7	311.3	489.8	706.5	963.4	–	–	–	–
0.7	31.5	132.9	305.4	547.7	854.3	1238.1	1691.4	–	–	–	–
1.0	45.0	190.0	435.0	780.0	1225.0	–	–	–	–	–	–

Table 7 Average number of vertices in grid graphs of size n

n	25	36	49	64	81	100	121	144	169	196	225
Vertices	40	60	84	112	144	180	220	264	312	364	420

5.2 Experimental results

According to Seipp, the number of extreme points in a MOST problem is bounded by $2q^{p-1}$, where q is the number of separating hyperplanes that can divide the weight region [33]. The author estimated that the number of hyperplanes is at most m^2, where $m = |E|$. However, the number of separating hyperplanes is at most $m(m-1)/2$, as each edge can only be swapped with the other $m-1$ edges, and amongst each pair of edges, independently of which edge is going in/out of the tree, only one separated hyperplane is generated. Therefore, the upper bound on the number of extreme points of the tri-objective MOST problem is $m^4 - 2m^3 + 2m^2$.

Figure 6 shows a comparison between the upper bound on the number of extreme points and the number of points obtained from our experiments for two different types of graphs (density and complete graphs). The number of solutions obtained is considerably smaller than the theoretical upper bound. Note that our algorithm obtains first each extreme tree and only thereafter it obtains the corresponding indifference region for the obtained tree. In contrast, in Seipp [33], the candidate indifference regions are built, and then, the extreme optimal solution associated to each region are calculated. This might result in the merge of several regions. As a consequence, the number of regions constructed and, consequently, the number of parametric spanning tree problems that the algorithm proposed in [33] has to solve can be as large as the maximum theoretical number of indifference regions. Thus, the large difference between the number of regions obtained by our algorithm and the maximum number of possible regions suggests that an algorithm that computes firstly the set of indifference regions might generate a large amount of regions

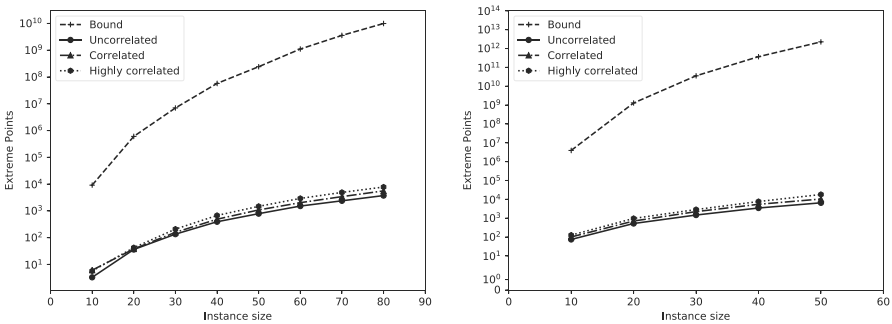

Fig. 6 Maximum number of extreme points and the number of extreme points obtained for density graphs with $d = 0.1$ (left), and for complete graphs (right)

leading to the same solution. That means that such an algorithm would spend a considerable amount of time recalculating spanning trees that were already found.

In the following, we present experimental results on the average CPU-time of our approach on the different types of instances. "Appendix B" contains three tables that summarize the average CPU-time results obtained for density, complete and grid graphs, respectively. Each table shows the number of vertices, correlation, density, average number of edges, average number of extreme points and average CPU-time in seconds for 30 runs on different instances per each setting. Note that the average number of supported spanning trees and extreme trees are omitted from the results as such number is almost the same in most cases. The reason for such a similarity may be related to the wide range of edge costs considered in our setting ([1, 1000]), which would unlikely lead to the generation of identical edges and, consequently, to increase the number of extreme or supported spanning trees in comparison to the number of extreme points.

Table 8 shows the effect of increasing the number of vertices and the number of edges on the running-time of our approach. Clearly, as the instance size grows, the algorithm requires more time to terminate. Table 9 shows the effect of changing the correlation value. Our results suggests that the correlation parameter has some impact on the performance of our approach.

6 Conclusions

The minimum spanning tree problem is a combinatorial optimization problem used in a wide range of different applications. The majority of real-life problems have often multiple conflicting objectives. These leads to the multiple objective spanning tree problem, which goal is to find the set of efficient spanning trees. This problem as been tackled in the majority for the bi-objective case, where several approaches have been developed. The majority of the approaches developed to generate the set of efficient trees are based on a two phases procedure, where in the first phase a pool of "good" candidates is calculated, usually the set of supported trees, to then be used as a bound to prune the search space. Literature is scarce on scenarios with three or more objectives, where few solutions have been proposed. To the best of our knowledge, the main works still do not exploit this reduction of the search space, possibly because either the efficient generation of the set of supported trees for that specific problem has not been properly studied in the literature, or the algorithms proposed have not been implemented or tested.

Table 8 Average CPU-time in seconds to solve uncorrelated instances for different size and density values

d	Vertices							
	10	20	30	40	50	60	70	80
0.1	0.0	0.0	0.2	2.1	10.6	43.0	125.7	325.2
0.4	0.0	0.4	6.4	47.0	219.7	714.2	2069.4	–
0.7	0.0	1.1	19.0	132.6	544.8	1814.5	–	–
1.0	0.0	2.4	30.8	241.0	950.3	–	–	–

Table 9 Average CPU-time in seconds to solve instances for different size and correlation values

Correlation	Vertices				
	10	20	30	40	50
0.0	0.0	2.5	30.8	241.0	950.3
− 0.4	0.0	3.5	48.3	333.3	1384.3
− 0.8	0.0	4.2	74.6	453.3	1944.8

In this article, we propose an algorithmic framework to calculate the set of supported spanning trees. We experimentally tested the algorithm on severel tri-objective spanning tree problem instances. The algorithm takes advantage of the connectedness properties of the set of supported solutions. It calculates this set of trees by generating the indifference regions associated to each supported spanning tree, and by searching in the boundaries of that region for neighbor trees that are also supported. The algorithm terminates when the union of the calculated indifference regions is equivalent to the entire weight set region. Our experimental results shows that our algorithm finds the set of supported spanning trees in a reasonable amount of time on a wide range of different instances. These results suggest that our approach can be used as a first phase within two-phase algorithmic frameworks for this problem.

Acknowledgements This work is financed by national funds through the FCT Foundation for Science and Technology, I.P. within the scope of the Project CISUC UID/CEC/00326/2020. Pedro Correia acknowledges the Portuguese funding institution FCT (Grant SFRH/BD/91647/2012). José Rui Figueira acknowledges the support of FCT Grant SFRH/BSAB/139892/2018 under POCH program and to the DOME (Discrete Optimization Methods for Energy management) FCT Research Project (Ref: PTDC/CCI-COM/31198/2017).

Appendix A: All efficient spanning trees and non-dominated points

See the Table 10.

Table 10 All efficient spanning trees for the numerical example

	P	N
1	$T^{1,1} = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{4,7\},\{6,7\}\}$	$z^1 = (13, 17, 11)$
2	$T^{1,2} = \{\{1,2\},\{2,3\},\{3,5\},\{2,6\},\{6,7\},\{4,7\}\}$	
3	$T^{1,3} = \{\{1,2\},\{2,3\},\{3,5\},\{3,6\},\{6,7\},\{4,7\}\}$	
4	$T^{1,4} = \{\{1,2\},\{2,3\},\{3,5\},\{5,7\},\{4,7\},\{6,7\}\}$	
5	$T^{2,1} = \{\{1,2\},\{2,3\},\{2,4\},\{3,5\},\{4,7\},\{6,7\}\}$	$z^2 = (14, 15, 16)$
6	$T^{2,2} = \{\{1,2\},\{2,3\},\{3,5\},\{5,6\},\{6,7\},\{4,7\}\}$	
7	$T^{2,3} = \{\{1,2\},\{2,3\},\{3,5\},\{3,7\},\{4,7\},\{6,7\}\}$	
8	$T^{3,1} = \{\{1,2\},\{2,3\},\{3,5\},\{1,7\},\{4,7\},\{6,7\}\}$	$z^3 = (18, 12, 15)$
9	$T^{3,2} = \{\{1,2\},\{2,3\},\{3,5\},\{2,7\},\{4,7\},\{6,7\}\}$	
10	$T^{4,1} = \{\{1,2\},\{2,3\},\{3,5\},\{4,5\},\{4,7\},\{6,7\}\}$	$z^4 = (21, 11, 11)$
11	$T^{4,2} = \{\{1,2\},\{2,3\},\{3,5\},\{1,6\},\{6,7\},\{4,7\}\}$	
12	$T^{5,1} = \{\{1,2\},\{2,3\},\{3,5\},\{4,5\},\{1,6\},\{4,7\}\}$	$z^5 = (30, 10, 10)$
13	$T^{5,2} = \{\{1,2\},\{2,3\},\{3,5\},\{4,5\},\{1,6\},\{6,7\}\}$	
14	$T^{5,3} = \{\{1,2\},\{2,3\},\{1,6\},\{6,7\},\{4,7\},\{4,5\}\}$	
15	$T^{5,4} = \{\{1,2\},\{1,6\},\{6,7\},\{4,7\},\{4,5\},\{3,5\}\}$	
16	$T^{5,5} = \{\{1,6\},\{6,7\},\{4,7\},\{4,5\},\{3,5\},\{2,3\}\}$	
17	$T^{6,1} = \{\{1,2\},\{1,4\},\{2,6\},\{3,6\},\{3,5\},\{5,7\}\}$	$z^6 = (16, 32, 8)$
18	$T^{6,2} = \{\{1,2\},\{1,4\},\{2,6\},\{3,6\},\{4,7\},\{5,7\}\}$	
19	$T^{6,3} = \{\{1,2\},\{1,4\},\{2,6\},\{3,6\},\{6,7\},\{5,7\}\}$	
20	$T^{6,4} = \{\{1,4\},\{4,7\},\{5,7\},\{3,5\},\{3,6\},\{2,6\}\}$	
21	$T^{6,5} = \{\{1,4\},\{4,7\},\{5,7\},\{6,7\},\{2,6\},\{3,6\}\}$	
22	$T^{7,1} = \{\{1,4\},\{4,5\},\{1,6\},\{2,6\},\{3,6\},\{5,7\}\}$	$z^7 = (34, 30, 6)$
23	$T^{8,1} = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{2,6\},\{4,7\}\}$	$z^8 = (14, 22, 10)$
24	$T^{8,2} = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{2,6\},\{6,7\}\}$	
25	$T^{8,3} = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{3,6\},\{4,7\}\}$	
26	$T^{8,4} = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{3,6\},\{6,7\}\}$	
27	$T^{8,5} = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{5,7\},\{6,7\}\}$	
28	$T^{8,6} = \{\{1,2\},\{2,3\},\{1,4\},\{4,7\},\{5,7\},\{6,7\}\}$	
29	$T^{8,7} = \{\{1,2\},\{2,3\},\{3,5\},\{2,6\},\{5,7\},\{4,7\}\}$	
30	$T^{8,8} = \{\{1,2\},\{2,3\},\{3,5\},\{3,6\},\{5,7\},\{4,7\}\}$	
31	$T^{8,9} = \{\{1,2\},\{2,3\},\{2,6\},\{6,7\},\{4,7\},\{5,7\}\}$	
32	$T^{8,10} = \{\{1,2\},\{2,3\},\{3,6\},\{6,7\},\{4,7\},\{5,7\}\}$	
33	$T^{8,11} = \{\{1,2\},\{1,4\},\{4,7\},\{5,7\},\{3,5\},\{6,7\}\}$	
34	$T^{8,12} = \{\{1,2\},\{1,4\},\{4,7\},\{6,7\},\{3,6\},\{3,5\}\}$	
35	$T^{8,13} = \{\{1,2\},\{2,6\},\{3,6\},\{3,5\},\{6,7\},\{4,7\}\}$	
36	$T^{8,14} = \{\{1,2\},\{2,6\},\{6,7\},\{4,7\},\{5,7\},\{3,5\}\}$	
37	$T^{8,15} = \{\{1,4\},\{4,7\},\{5,7\},\{3,5\},\{2,3\},\{6,7\}\}$	
38	$T^{8,16} = \{\{1,4\},\{4,7\},\{6,7\},\{2,6\},\{2,3\},\{3,5\}\}$	
39	$T^{8,17} = \{\{1,4\},\{4,7\},\{6,7\},\{3,6\},\{2,3\},\{3,5\}\}$	

Table 10 continued

	P	N
40	$T^{9,1} = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{2,6\},\{5,7\}\}$	$z^9 = (15,27,9)$
41	$T^{9,2} = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{3,6\},\{5,7\}\}$	
42	$T^{9,3} = \{\{1,2\},\{2,3\},\{1,4\},\{2,6\},\{4,7\},\{5,7\}\}$	
43	$T^{9,4} = \{\{1,2\},\{2,3\},\{1,4\},\{2,6\},\{6,7\},\{5,7\}\}$	
44	$T^{9,5} = \{\{1,2\},\{2,3\},\{1,4\},\{3,6\},\{4,7\},\{5,7\}\}$	
45	$T^{9,6} = \{\{1,2\},\{2,3\},\{1,4\},\{3,6\},\{6,7\},\{5,7\}\}$	
46	$T^{9,7} = \{\{1,2\},\{1,4\},\{2,6\},\{3,6\},\{3,5\},\{4,7\}\}$	
47	$T^{9,8} = \{\{1,2\},\{1,4\},\{2,6\},\{3,6\},\{3,5\},\{6,7\}\}$	
48	$T^{9,9} = \{\{1,2\},\{1,4\},\{2,6\},\{4,7\},\{5,7\},\{3,5\}\}$	
49	$T^{9,10} = \{\{1,2\},\{1,4\},\{2,6\},\{6,7\},\{5,7\},\{3,5\}\}$	
50	$T^{9,11} = \{\{1,2\},\{1,4\},\{4,7\},\{5,7\},\{3,5\},\{3,6\}\}$	
51	$T^{9,12} = \{\{1,2\},\{1,4\},\{4,7\},\{5,7\},\{6,7\},\{3,6\}\}$	
52	$T^{9,13} = \{\{1,2\},\{2,6\},\{3,6\},\{3,5\},\{5,7\},\{4,7\}\}$	
53	$T^{9,14} = \{\{1,2\},\{2,6\},\{3,6\},\{6,7\},\{4,7\},\{5,7\}\}$	
54	$T^{9,15} = \{\{1,4\},\{4,7\},\{5,7\},\{3,5\},\{2,3\},\{2,6\}\}$	
55	$T^{9,16} = \{\{1,4\},\{4,7\},\{5,7\},\{3,5\},\{2,3\},\{3,6\}\}$	
56	$T^{9,17} = \{\{1,4\},\{4,7\},\{5,7\},\{3,5\},\{6,7\},\{2,6\}\}$	
57	$T^{9,18} = \{\{1,4\},\{4,7\},\{5,7\},\{6,7\},\{2,6\},\{2,3\}\}$	
58	$T^{9,19} = \{\{1,4\},\{4,7\},\{5,7\},\{6,7\},\{3,6\},\{2,3\}\}$	
59	$T^{9,20} = \{\{1,4\},\{4,7\},\{6,7\},\{2,6\},\{3,6\},\{3,5\}\}$	
60	$T^{10,1} = \{\{1,2\},\{2,3\},\{1,4\},\{4,5\},\{1,6\},\{4,7\}\}$	$z^{10} = (31,15,9)$
61	$T^{10,2} = \{\{1,2\},\{2,3\},\{1,4\},\{4,5\},\{1,6\},\{6,7\}\}$	
62	$T^{10,3} = \{\{1,2\},\{2,3\},\{3,5\},\{4,5\},\{1,6\},\{5,7\}\}$	
63	$T^{10,4} = \{\{1,2\},\{2,3\},\{1,6\},\{6,7\},\{5,7\},\{4,5\}\}$	
64	$T^{10,5} = \{\{1,2\},\{1,4\},\{4,5\},\{3,5\},\{1,6\},\{4,7\}\}$	
65	$T^{10,6} = \{\{1,2\},\{1,4\},\{4,5\},\{3,5\},\{1,6\},\{6,7\}\}$	
66	$T^{10,7} = \{\{1,2\},\{1,6\},\{3,6\},\{3,5\},\{4,5\},\{4,7\}\}$	
67	$T^{10,8} = \{\{1,2\},\{1,6\},\{3,6\},\{3,5\},\{4,5\},\{6,7\}\}$	
68	$T^{10,9} = \{\{1,2\},\{1,6\},\{3,6\},\{6,7\},\{4,7\},\{4,5\}\}$	
69	$T^{10,10} = \{\{1,2\},\{1,6\},\{6,7\},\{5,7\},\{3,5\},\{4,5\}\}$	
70	$T^{10,11} = \{\{1,4\},\{4,5\},\{3,5\},\{2,3\},\{1,6\},\{4,7\}\}$	
71	$T^{10,12} = \{\{1,4\},\{4,5\},\{3,5\},\{2,3\},\{1,6\},\{6,7\}\}$	
72	$T^{10,13} = \{\{1,6\},\{2,6\},\{2,3\},\{3,5\},\{4,5\},\{4,7\}\}$	
73	$T^{10,14} = \{\{1,6\},\{2,6\},\{2,3\},\{3,5\},\{4,5\},\{6,7\}\}$	
74	$T^{10,15} = \{\{1,6\},\{2,6\},\{2,3\},\{6,7\},\{4,7\},\{4,5\}\}$	
75	$T^{10,16} = \{\{1,6\},\{2,6\},\{6,7\},\{4,7\},\{4,5\},\{3,5\}\}$	
76	$T^{10,17} = \{\{1,6\},\{3,6\},\{2,3\},\{3,5\},\{4,5\},\{4,7\}\}$	
77	$T^{10,18} = \{\{1,6\},\{3,6\},\{2,3\},\{3,5\},\{4,5\},\{6,7\}\}$	
78	$T^{10,19} = \{\{1,6\},\{3,6\},\{2,3\},\{6,7\},\{4,7\},\{4,5\}\}$	
79	$T^{10,20} = \{\{1,6\},\{6,7\},\{5,7\},\{3,5\},\{2,3\},\{4,5\}\}$	

Table 10 continued

	P	N
80	$T^{11,1} = \{\{1,2\},\{2,3\},\{1,4\},\{4,5\},\{1,6\},\{5,7\}\}$	$z^{11} = (32,20,8)$
81	$T^{11,2} = \{\{1,2\},\{1,4\},\{4,5\},\{3,5\},\{1,6\},\{5,7\}\}$	
82	$T^{11,3} = \{\{1,2\},\{1,4\},\{4,5\},\{1,6\},\{3,6\},\{4,7\}\}$	
83	$T^{11,4} = \{\{1,2\},\{1,4\},\{4,5\},\{1,6\},\{3,6\},\{6,7\}\}$	
84	$T^{11,5} = \{\{1,2\},\{1,6\},\{3,6\},\{3,5\},\{4,5\},\{5,7\}\}$	
85	$T^{11,6} = \{\{1,2\},\{1,6\},\{3,6\},\{6,7\},\{5,7\},\{4,5\}\}$	
86	$T^{11,7} = \{\{1,4\},\{4,5\},\{3,5\},\{2,3\},\{1,6\},\{5,7\}\}$	
87	$T^{11,8} = \{\{1,4\},\{4,5\},\{3,5\},\{1,6\},\{2,6\},\{4,7\}\}$	
88	$T^{11,9} = \{\{1,4\},\{4,5\},\{3,5\},\{1,6\},\{2,6\},\{6,7\}\}$	
89	$T^{11,10} = \{\{1,4\},\{4,5\},\{1,6\},\{2,6\},\{2,3\},\{4,7\}\}$	
90	$T^{11,11} = \{\{1,4\},\{4,5\},\{1,6\},\{2,6\},\{2,3\},\{6,7\}\}$	
91	$T^{11,12} = \{\{1,4\},\{4,5\},\{1,6\},\{3,6\},\{2,3\},\{4,7\}\}$	
92	$T^{11,13} = \{\{1,4\},\{4,5\},\{1,6\},\{3,6\},\{2,3\},\{6,7\}\}$	
93	$T^{11,14} = \{\{1,6\},\{2,6\},\{2,3\},\{3,5\},\{4,5\},\{5,7\}\}$	
94	$T^{11,15} = \{\{1,6\},\{2,6\},\{2,3\},\{6,7\},\{5,7\},\{4,5\}\}$	
95	$T^{11,16} = \{\{1,6\},\{2,6\},\{3,6\},\{3,5\},\{4,5\},\{4,7\}\}$	
96	$T^{11,17} = \{\{1,6\},\{2,6\},\{3,6\},\{3,5\},\{4,5\},\{6,7\}\}$	
97	$T^{11,18} = \{\{1,6\},\{2,6\},\{3,6\},\{6,7\},\{4,7\},\{4,5\}\}$	
98	$T^{11,19} = \{\{1,6\},\{2,6\},\{6,7\},\{5,7\},\{3,5\},\{4,5\}\}$	
99	$T^{11,20} = \{\{1,6\},\{3,6\},\{2,3\},\{3,5\},\{4,5\},\{5,7\}\}$	
100	$T^{11,21} = \{\{1,6\},\{3,6\},\{2,3\},\{6,7\},\{5,7\},\{4,5\}\}$	
101	$T^{12,1} = \{\{1,2\},\{1,4\},\{4,5\},\{1,6\},\{3,6\},\{5,7\}\}$	$z^{12} = (33,25,7)$
102	$T^{12,2} = \{\{1,4\},\{4,5\},\{3,5\},\{1,6\},\{2,6\},\{5,7\}\}$	
103	$T^{12,3} = \{\{1,4\},\{4,5\},\{1,6\},\{2,6\},\{2,3\},\{5,7\}\}$	
104	$T^{12,4} = \{\{1,4\},\{4,5\},\{1,6\},\{2,6\},\{3,6\},\{4,7\}\}$	
105	$T^{12,5} = \{\{1,4\},\{4,5\},\{1,6\},\{2,6\},\{3,6\},\{6,7\}\}$	
106	$T^{12,6} = \{\{1,4\},\{4,5\},\{1,6\},\{3,6\},\{2,3\},\{5,7\}\}$	
107	$T^{12,7} = \{\{1,6\},\{2,6\},\{3,6\},\{3,5\},\{4,5\},\{5,7\}\}$	
108	$T^{12,8} = \{\{1,6\},\{2,6\},\{3,6\},\{6,7\},\{5,7\},\{4,5\}\}$	
109	$T^{13,1} = \{\{1,2\},\{1,4\},\{4,5\},\{2,6\},\{3,6\},\{5,7\}\}$	$z^{13} = (25,31,7)$
110	$T^{13,2} = \{\{1,4\},\{4,5\},\{3,5\},\{3,6\},\{2,6\},\{5,7\}\}$	
111	$T^{13,3} = \{\{1,4\},\{4,5\},\{5,7\},\{6,7\},\{2,6\},\{3,6\}\}$	
112	$T^{13,4} = \{\{1,4\},\{1,6\},\{2,6\},\{3,6\},\{3,5\},\{5,7\}\}$	
113	$T^{13,5} = \{\{1,4\},\{1,6\},\{2,6\},\{3,6\},\{4,7\},\{5,7\}\}$	
114	$T^{13,6} = \{\{1,4\},\{1,6\},\{2,6\},\{3,6\},\{6,7\},\{5,7\}\}$	

Table 10 continued

	P	N
115	$T^{14,1} = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{1,6\},\{4,7\}\}$	$z^{14} = (22, 16, 10)$
116	$T^{14,2} = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{1,6\},\{6,7\}\}$	
117	$T^{14,3} = \{\{1,2\},\{2,3\},\{1,4\},\{4,5\},\{4,7\},\{6,7\}\}$	
118	$T^{14,4} = \{\{1,2\},\{2,3\},\{3,5\},\{4,5\},\{2,6\},\{4,7\}\}$	
119	$T^{14,5} = \{\{1,2\},\{2,3\},\{3,5\},\{4,5\},\{2,6\},\{6,7\}\}$	
120	$T^{14,6} = \{\{1,2\},\{2,3\},\{3,5\},\{4,5\},\{3,6\},\{4,7\}\}$	
121	$T^{14,7} = \{\{1,2\},\{2,3\},\{3,5\},\{4,5\},\{3,6\},\{6,7\}\}$	
122	$T^{14,8} = \{\{1,2\},\{2,3\},\{3,5\},\{4,5\},\{5,7\},\{6,7\}\}$	
123	$T^{14,9} = \{\{1,2\},\{2,3\},\{3,5\},\{1,6\},\{5,7\},\{4,7\}\}$	
124	$T^{14,10} = \{\{1,2\},\{2,3\},\{1,6\},\{6,7\},\{4,7\},\{5,7\}\}$	
125	$T^{14,11} = \{\{1,2\},\{2,3\},\{2,6\},\{6,7\},\{4,7\},\{4,5\}\}$	
126	$T^{14,12} = \{\{1,2\},\{2,3\},\{3,6\},\{6,7\},\{4,7\},\{4,5\}\}$	
127	$T^{14,13} = \{\{1,2\},\{1,4\},\{4,5\},\{3,5\},\{4,7\},\{6,7\}\}$	
128	$T^{14,14} = \{\{1,2\},\{1,6\},\{3,6\},\{3,5\},\{6,7\},\{4,7\}\}$	
129	$T^{14,15} = \{\{1,2\},\{1,6\},\{6,7\},\{4,7\},\{5,7\},\{3,5\}\}$	
130	$T^{14,16} = \{\{1,2\},\{2,6\},\{6,7\},\{4,7\},\{4,5\},\{3,5\}\}$	
131	$T^{14,17} = \{\{1,4\},\{4,5\},\{3,5\},\{2,3\},\{4,7\},\{6,7\}\}$	
132	$T^{14,18} = \{\{1,6\},\{2,6\},\{2,3\},\{3,5\},\{6,7\},\{4,7\}\}$	
133	$T^{14,19} = \{\{1,6\},\{3,6\},\{2,3\},\{3,5\},\{6,7\},\{4,7\}\}$	
134	$T^{14,20} = \{\{1,6\},\{6,7\},\{4,7\},\{5,7\},\{3,5\},\{2,3\}\}$	
135	$T^{15,1} = \{\{1,2\},\{2,3\},\{1,4\},\{3,5\},\{1,6\},\{5,7\}\}$	$z^{15} = (23, 21, 9)$
136	$T^{15,2} = \{\{1,2\},\{2,3\},\{1,4\},\{4,5\},\{2,6\},\{4,7\}\}$	
137	$T^{15,3} = \{\{1,2\},\{2,3\},\{1,4\},\{4,5\},\{2,6\},\{6,7\}\}$	
138	$T^{15,4} = \{\{1,2\},\{2,3\},\{1,4\},\{4,5\},\{3,6\},\{4,7\}\}$	
139	$T^{15,5} = \{\{1,2\},\{2,3\},\{1,4\},\{4,5\},\{3,6\},\{6,7\}\}$	
140	$T^{15,6} = \{\{1,2\},\{2,3\},\{1,4\},\{4,5\},\{5,7\},\{6,7\}\}$	
141	$T^{15,7} = \{\{1,2\},\{2,3\},\{1,4\},\{1,6\},\{4,7\},\{5,7\}\}$	
142	$T^{15,8} = \{\{1,2\},\{2,3\},\{1,4\},\{1,6\},\{6,7\},\{5,7\}\}$	
143	$T^{15,9} = \{\{1,2\},\{2,3\},\{3,5\},\{4,5\},\{2,6\},\{5,7\}\}$	
144	$T^{15,10} = \{\{1,2\},\{2,3\},\{3,5\},\{4,5\},\{3,6\},\{5,7\}\}$	
145	$T^{15,11} = \{\{1,2\},\{2,3\},\{2,6\},\{6,7\},\{5,7\},\{4,5\}\}$	
146	$T^{15,12} = \{\{1,2\},\{2,3\},\{3,6\},\{6,7\},\{5,7\},\{4,5\}\}$	
147	$T^{15,13} = \{\{1,2\},\{1,4\},\{4,5\},\{3,5\},\{2,6\},\{4,7\}\}$	
148	$T^{15,14} = \{\{1,2\},\{1,4\},\{4,5\},\{3,5\},\{2,6\},\{6,7\}\}$	
149	$T^{15,15} = \{\{1,2\},\{1,4\},\{4,5\},\{3,5\},\{3,6\},\{4,7\}\}$	
150	$T^{15,16} = \{\{1,2\},\{1,4\},\{4,5\},\{3,5\},\{3,6\},\{6,7\}\}$	
151	$T^{15,17} = \{\{1,2\},\{1,4\},\{4,5\},\{3,5\},\{5,7\},\{6,7\}\}$	
152	$T^{15,18} = \{\{1,2\},\{1,4\},\{4,5\},\{4,7\},\{6,7\},\{3,6\}\}$	
153	$T^{15,19} = \{\{1,2\},\{1,4\},\{1,6\},\{3,6\},\{3,5\},\{4,7\}\}$	

Table 10 continued

	P	N
154	$T^{15,20} = \{\{1,2\},\{1,4\},\{1,6\},\{3,6\},\{3,5\},\{6,7\}\}$	
155	$T^{15,21} = \{\{1,2\},\{1,4\},\{1,6\},\{4,7\},\{5,7\},\{3,5\}\}$	
156	$T^{15,22} = \{\{1,2\},\{1,4\},\{1,6\},\{6,7\},\{5,7\},\{3,5\}\}$	
157	$T^{15,23} = \{\{1,2\},\{1,6\},\{3,6\},\{3,5\},\{5,7\},\{4,7\}\}$	
158	$T^{15,24} = \{\{1,2\},\{1,6\},\{3,6\},\{6,7\},\{4,7\},\{5,7\}\}$	
159	$T^{15,25} = \{\{1,2\},\{2,6\},\{3,6\},\{3,5\},\{4,5\},\{4,7\}\}$	
160	$T^{15,26} = \{\{1,2\},\{2,6\},\{3,6\},\{3,5\},\{4,5\},\{6,7\}\}$	
161	$T^{15,27} = \{\{1,2\},\{2,6\},\{3,6\},\{6,7\},\{4,7\},\{4,5\}\}$	
162	$T^{15,28} = \{\{1,2\},\{2,6\},\{6,7\},\{5,7\},\{3,5\},\{4,5\}\}$	
163	$T^{15,29} = \{\{1,4\},\{4,5\},\{3,5\},\{2,3\},\{2,6\},\{4,7\}\}$	
164	$T^{15,30} = \{\{1,4\},\{4,5\},\{3,5\},\{2,3\},\{2,6\},\{6,7\}\}$	
165	$T^{15,31} = \{\{1,4\},\{4,5\},\{3,5\},\{2,3\},\{3,6\},\{4,7\}\}$	
166	$T^{15,32} = \{\{1,4\},\{4,5\},\{3,5\},\{2,3\},\{3,6\},\{6,7\}\}$	
167	$T^{15,33} = \{\{1,4\},\{4,5\},\{3,5\},\{2,3\},\{5,7\},\{6,7\}\}$	
168	$T^{15,34} = \{\{1,4\},\{4,5\},\{3,5\},\{4,7\},\{6,7\},\{2,6\}\}$	
169	$T^{15,35} = \{\{1,4\},\{4,5\},\{4,7\},\{6,7\},\{2,6\},\{2,3\}\}$	
170	$T^{15,36} = \{\{1,4\},\{4,5\},\{4,7\},\{6,7\},\{3,6\},\{2,3\}\}$	
171	$T^{15,37} = \{\{1,4\},\{1,6\},\{2,6\},\{2,3\},\{3,5\},\{4,7\}\}$	
172	$T^{15,38} = \{\{1,4\},\{1,6\},\{2,6\},\{2,3\},\{3,5\},\{6,7\}\}$	
173	$T^{15,39} = \{\{1,4\},\{1,6\},\{3,6\},\{2,3\},\{3,5\},\{4,7\}\}$	
174	$T^{15,40} = \{\{1,4\},\{1,6\},\{3,6\},\{2,3\},\{3,5\},\{6,7\}\}$	
175	$T^{15,41} = \{\{1,4\},\{1,6\},\{4,7\},\{5,7\},\{3,5\},\{2,3\}\}$	
176	$T^{15,42} = \{\{1,4\},\{1,6\},\{6,7\},\{5,7\},\{3,5\},\{2,3\}\}$	
177	$T^{15,43} = \{\{1,6\},\{2,6\},\{2,3\},\{3,5\},\{5,7\},\{4,7\}\}$	
178	$T^{15,44} = \{\{1,6\},\{2,6\},\{2,3\},\{6,7\},\{4,7\},\{5,7\}\}$	
179	$T^{15,45} = \{\{1,6\},\{2,6\},\{3,6\},\{3,5\},\{6,7\},\{4,7\}\}$	
180	$T^{15,46} = \{\{1,6\},\{2,6\},\{6,7\},\{4,7\},\{5,7\},\{3,5\}\}$	
181	$T^{15,47} = \{\{1,6\},\{3,6\},\{2,3\},\{3,5\},\{5,7\},\{4,7\}\}$	
182	$T^{15,48} = \{\{1,6\},\{3,6\},\{2,3\},\{6,7\},\{4,7\},\{5,7\}\}$	
183	$T^{16,1} = \{\{1,2\},\{2,3\},\{1,4\},\{4,5\},\{2,6\},\{5,7\}\}$	$z^{16} = (24,26,8)$
184	$T^{16,2} = \{\{1,2\},\{2,3\},\{1,4\},\{4,5\},\{3,6\},\{5,7\}\}$	
185	$T^{16,3} = \{\{1,2\},\{1,4\},\{4,5\},\{3,5\},\{2,6\},\{5,7\}\}$	
186	$T^{16,4} = \{\{1,2\},\{1,4\},\{4,5\},\{3,5\},\{3,6\},\{5,7\}\}$	
187	$T^{16,5} = \{\{1,2\},\{1,4\},\{4,5\},\{2,6\},\{3,6\},\{4,7\}\}$	
188	$T^{16,6} = \{\{1,2\},\{1,4\},\{4,5\},\{2,6\},\{3,6\},\{6,7\}\}$	
189	$T^{16,7} = \{\{1,2\},\{1,4\},\{4,5\},\{5,7\},\{6,7\},\{3,6\}\}$	
190	$T^{16,8} = \{\{1,2\},\{1,4\},\{1,6\},\{3,6\},\{3,5\},\{5,7\}\}$	
191	$T^{16,9} = \{\{1,2\},\{1,4\},\{1,6\},\{3,6\},\{4,7\},\{5,7\}\}$	
192	$T^{16,10} = \{\{1,2\},\{1,4\},\{1,6\},\{3,6\},\{6,7\},\{5,7\}\}$	

Table 10 continued

	P	N
193	$T^{16,11} = \{\{1,2\},\{2,6\},\{3,6\},\{3,5\},\{4,5\},\{5,7\}\}$	
194	$T^{16,12} = \{\{1,2\},\{2,6\},\{3,6\},\{6,7\},\{5,7\},\{4,5\}\}$	
195	$T^{16,13} = \{\{1,4\},\{4,5\},\{3,5\},\{2,3\},\{2,6\},\{5,7\}\}$	
196	$T^{16,14} = \{\{1,4\},\{4,5\},\{3,5\},\{2,3\},\{3,6\},\{5,7\}\}$	
197	$T^{16,15} = \{\{1,4\},\{4,5\},\{3,5\},\{3,6\},\{2,6\},\{4,7\}\}$	
198	$T^{16,16} = \{\{1,4\},\{4,5\},\{3,5\},\{3,6\},\{2,6\},\{6,7\}\}$	
199	$T^{16,17} = \{\{1,4\},\{4,5\},\{3,5\},\{5,7\},\{6,7\},\{2,6\}\}$	
200	$T^{16,18} = \{\{1,4\},\{4,5\},\{4,7\},\{6,7\},\{2,6\},\{3,6\}\}$	
201	$T^{16,19} = \{\{1,4\},\{4,5\},\{5,7\},\{6,7\},\{2,6\},\{2,3\}\}$	
202	$T^{16,20} = \{\{1,4\},\{4,5\},\{5,7\},\{6,7\},\{3,6\},\{2,3\}\}$	
203	$T^{16,21} = \{\{1,4\},\{1,6\},\{2,6\},\{2,3\},\{3,5\},\{5,7\}\}$	
204	$T^{16,22} = \{\{1,4\},\{1,6\},\{2,6\},\{2,3\},\{4,7\},\{5,7\}\}$	
205	$T^{16,23} = \{\{1,4\},\{1,6\},\{2,6\},\{2,3\},\{6,7\},\{5,7\}\}$	
206	$T^{16,24} = \{\{1,4\},\{1,6\},\{2,6\},\{3,6\},\{3,5\},\{4,7\}\}$	
207	$T^{16,25} = \{\{1,4\},\{1,6\},\{2,6\},\{3,6\},\{3,5\},\{6,7\}\}$	
208	$T^{16,26} = \{\{1,4\},\{1,6\},\{2,6\},\{4,7\},\{5,7\},\{3,5\}\}$	
209	$T^{16,27} = \{\{1,4\},\{1,6\},\{2,6\},\{6,7\},\{5,7\},\{3,5\}\}$	
210	$T^{16,28} = \{\{1,4\},\{1,6\},\{3,6\},\{2,3\},\{3,5\},\{5,7\}\}$	
211	$T^{16,29} = \{\{1,4\},\{1,6\},\{3,6\},\{2,3\},\{4,7\},\{5,7\}\}$	
212	$T^{16,30} = \{\{1,4\},\{1,6\},\{3,6\},\{2,3\},\{6,7\},\{5,7\}\}$	
213	$T^{16,31} = \{\{1,6\},\{2,6\},\{3,6\},\{3,5\},\{5,7\},\{4,7\}\}$	
214	$T^{16,32} = \{\{1,6\},\{2,6\},\{3,6\},\{6,7\},\{4,7\},\{5,7\}\}$	

Appendix B: Experimental results

See the Tables 11 and 12.

Table 11 Number of supported trees, points, extreme points and CPU-time for density graphs 1

Vertices	Correlation	Density	Edges	Extreme points	CPU-time
10	0.0	0.1	10.3	3.3	0.0
		0.4	18.6	23.6	0.0
		0.7	32.0	51.8	0.0
		1.0	45.0	74.4	0.0
	− 0.4	0.1	11.0	6.1	0.0
		0.4	19.4	33.0	0.0
		0.7	31.8	65.9	0.0
		1.0	45.0	100.8	0.0
	− 0.8	0.1	10.4	5.7	0.0
		0.4	19.9	43.6	0.0
		0.7	31.2	83.0	0.0
		1.0	45.0	126.0	0.0
20	0.0	0.1	28.4	36.1	0.0
		0.4	74.7	229.1	0.3
		0.7	132.4	393.9	1.1
		1.0	190.0	521.8	2.4
	− 0.4	0.1	26.9	37.9	0.0
		0.4	77.2	329.1	1.2
		0.7	130.8	564.4	2.6
		1.0	190.0	706.2	3.5
	− 0.8	0.1	26.4	42.3	0.0
		0.4	77.0	432.8	1.4
		0.7	134.6	729.7	3.0
		1.0	190.0	968.3	4.1
30	0.0	0.1	51.9	135.1	0.2
		0.4	175.0	814.2	6.4
		0.7	304.7	1318.1	19.0
		1.0	435.0	1469.2	30.8
	− 0.4	0.1	50.8	158.1	0.2
		0.4	174.1	1075.9	10.1
		0.7	308.3	1825.0	33.5
		1.0	435.0	2226.9	48.3
	− 0.8	0.1	50.4	208.3	0.5
		0.4	173.6	1499.0	24.3
		0.7	304.8	2326.9	49.1
		1.0	435.0	2872.4	74.6

Table 11 continued

Vertices	Correlation	Density	Edges	Extreme points	CPU-time
40	0.0	0.1	87.6	392.6	2.1
		0.4	310.3	1792.3	47.0
		0.7	547.7	2850.7	132.6
		1.0	780.0	3468.1	241.0
	− 0.4	0.1	86.2	488.1	3.2
		0.4	309.3	2522.6	110.2
		0.7	544.5	4058.2	250.6
		1.0	780.0	5444.9	333.3
	− 0.8	0.1	85.5	674.3	5.3
		0.4	311.7	3439.1	220.1
		0.7	548.5	5582.3	341.9
		1.0	780.0	7645.5	453.3
50	0.0	0.1	125.3	787.3	10.6
		0.4	491.7	3657.1	219.7
		0.7	862.0	5214.2	544.8
		1.0	1225.0	6535.8	950.3
	− 0.4	0.1	128.6	1084.2	15.3
		0.4	489.4	5200.8	329.7
		0.7	857.8	8087.8	1149.3
		1.0	1225.0	10,410.4	1384.3
	− 0.8	0.1	129.5	1475.8	21.4
		0.4	489.4	7801.9	530.3
		0.7	858.2	16,646.0	1723.1
		1.0	1225.0	18,057.2	1944.8
60	0.0	0.1	183.0	1513.6	43.1
		0.4	700.8	5504.8	714.2
		0.7	1236.8	8791.3	1814.4

Table 12 Number of supported trees, points, extreme points and CPU-time for grid graphs

Vertices	Correlation	Density	Edges	Extreme points	CPU-time
25	0.0	1	40.0	92.4	0.1
	− 0.4	1	40.0	117.2	0.1
	− 0.8	1	40.0	156.7	0.2
36	0.0	1	60.0	195.8	0.5
	− 0.4	1	60.0	264.2	0.7
	− 0.8	1	60.0	344.9	1.0
49	0.0	1	84.0	389.4	2.7
	− 0.4	1	84.0	497.0	3.6
	− 0.8	1	84.0	705.0	5.4

Table 12 continued

Vertices	Correlation	Density	Edges	Extreme points	CPU-time
64	0.0	1	112.0	662.2	10.9
	− 0.4	1	112.0	841.9	14.3
	− 0.8	1	112.0	1202.8	19.7
81	0.0	1	144.0	1068.3	36.7
	− 0.4	1	144.0	1527.9	53.4
	− 0.8	1	144.0	2067.1	72.3
100	0.0	1	180.0	1680.8	109.3
	− 0.4	1	180.0	2340.4	149.4
	− 0.8	1	180.0	3346.9	206.7
121	0.0	1	220.0	2605.4	290.8
	− 0.4	1	220.0	3602.2	391.7
	− 0.8	1	220.0	5424.7	546.3
144	0.0	1	264.0	3711.6	688.7
	− 0.4	1	264.0	5065.8	936.9
	− 0.8	1	264.0	7829.3	1345.6
169	0.0	1	312.0	5144.9	1524.3
	− 0.4	1	312.0	7720.3	2265.4
	− 0.8	1	312.0	11,545.0	2445.2
196	0.0	1	364.0	6968.5	3315.9
	− 0.4	1	364.0	10,952.0	5052.2
	− 0.8	1	364.0	20,786.1	6023.8

References

1. Ackermann, W.: Zum Hilbertschen Aufbau der reellen Zahlen. Math. Ann. **99**, 118–133 (1928)
2. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Inc., Upper Saddle River (1993)
3. Alves, M.J., Costa, J.P.: Graphical exploration of the weight space in three-objective mixed integer linear programs. Eur. J. Oper. Res. **248**(1), 72–83 (2016)
4. Andersen, K.A., Jörnsten, K., Lind, M.: On bicriterion minimal spanning trees: an approximation. Comput. Oper. Res. **23**(12), 1171–1182 (1996)
5. Bökler, F., Mutzel, P.: Output-sensitive algorithms for enumerating the extreme nondominated points of multiobjective combinatorial optimization problems. Lect. Notes Comput. Sci. **9294**, 288–299 (2015)
6. Cayley, A.: A theorem on trees. Q. J. Math. **23**, 376–378 (1889)
7. Chazelle, B.: A minimum spanning tree algorithm with inverse-ackermann type complexity. J. ACM **47**(6), 1028–1047 (2000)
8. Chou, W., Kershenbaum, A.: A unified algorithm for designing multidrop teleprocessing networks. In: Proceedings of the Third ACM Symposium on Data Communications and Data Networks: Analysis and Design, DATACOMM '73, pp. 148–156. ACM, NY (1973)
9. Corley, H.W.: Efficient spanning trees. J. Optim. Theory Appl. **45**(3), 481–485 (1985)
10. Ehrgott, M.: Multicriteria optimization. In: Lecture Notes in Economics and Mathematical Systems, volume 491, 2nd edn. Springer, Berlin (2005)

11. Ehrgott, M., Klamroth, K.: Connectedness of efficient solutions in multiple criteria combinatorial optimization. Eur. J. Oper. Res. **97**(1), 159–166 (1997)
12. Eppstein, D.: Representing All Minimum Spanning Trees with Applications to Counting and Generation. Technical Report 95-50, University of California, Irvine, Department of Information and Computer Science,, California (1995)
13. Esau, L.R., Williams, K.C.: On teleprocessing system design: part ii a method for approximating the optimal network. IBM Syst. J. **5**(3), 142–147 (1966)
14. Figueira, J., Paquete, L., Simes, M.A.M., Vanderpooten, D.: Algorithmic improvements on dynamic programming for the bi-objective 0,1 knapsack problem. Comput. Optim. Appl. **56**(1), 97–111 (2013)
15. Geoffrion, A.M.: Proper efficiency and the theory of vector maximization. J. Math. Anal. Appl. **22**(3), 618–630 (1968)
16. Gorski, J.: Multiple Objective Optimization and its Implications to Single Objective Optimization Problems. Ph.D. Thesis, University of Wuppertal, Germany (2010)
17. Gorski, J., Klamroth, K., Ruzika, S.: Connectedness of efficient solutions in multiple objective combinatorial optimization. J. Optim. Theory Appl. **150**(3), 475–497 (2011)
18. Hamacher, H.W., Ruhe, G.: On spanning tree problems with multiple objectives. Ann. Oper. Res. **52**(4), 209–230 (1994)
19. Kapoor, S., Ramesh, H.: Algorithms for enumerating all spanning trees of undirected and weighted graphs. SIAM J. Comput. **24**(2), 247–265 (1995)
20. Karger, D.R., Klein, P.N., Tarjan, R.E.: A randomized linear-time algorithm to find minimum spanning trees. J. ACM **42**(2), 321–328 (1995)
21. Knowles, J.D., Corne, D.W.: A Comparison of Encodings and Algorithms for Multi-objective Minimum Spanning Tree Problems, volume 1, pp. 544–551. IEEE Press, Piscataway (2001)
22. Kruskal, J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem. Proc. Am. Math. Soc. **7**(1), 48–50 (1956)
23. Lacour, R.: Exact and Approximate Solving Approaches in Multi-objective Combinatorial Optimization, Application to the Minimum Weight Spanning Tree Problem. Ph.D. Thesis, Université Paris-Dauphine, France (2014)
24. Loberman, H., Weinberger, A.: Formal procedures for connecting terminals with a minimum total wire length. J. ACM **4**(4), 428–437 (1957)
25. Özpeynirci, Ö., Köksalan, M.: An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer programs. Manag. Sci. **56**, 2302–2315 (2010)
26. Prim, R.C.: Shortest connection networks and some generalizations. Bell Syst. Technol. J. **36**, 1389–1401 (1957)
27. Przybylski, A., Gandibleux, X., Ehrgott, M.: A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. INFORMS J. Comput. **22**(3), 371–386 (2009)
28. Pugliese, L.P., Guerriero, F., Santos, J.L.: Dynamic programming for spanning tree problems: application to the multi-objective case. Optim. Lett. **9**(3), 437–450 (2015)
29. Ramos, R.M., Alonso, S., Sicilia, J., González, C.: The problem of the optimal biobjective spanning tree. Eur. J. Oper. Res. **111**(3), 617–628 (1998)
30. Rossi, J.A., Heiser, R.S., King, N.S.: A Cost Analysis of Minimum Distance TV Networking for Broadcasting Medical Information. Rand Corporation, Santa Monica (1970)
31. Ruzika, S., Hamacher, H.W.: A Survey on Multiple Objective Minimum Spanning Tree Problems, pp. 104–116. Springer, Berlin (2009)
32. Saltman, R.G., Bolotsky, G.R., Ruthberg, Z.G.: Heuristic cost optimization of the federal telpak network. In: United States Department of Commerce. National Bureau of Standards. Technical note. U.S. National Bureau of Standards (1973)
33. Seipp, F.: On Adjacency, Cardinality, and Partial Dominance in Discrete Multiple Objective Optimization. Ph.D. Thesis, Technische Universität Kaiserslauern, Germany (2013)
34. 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)
35. Steiner, S., Radzik, T.: Computing all efficient solutions of the biobjective minimum spanning tree problem. Comput. Oper. Res. **35**(1), 198–211 (2008)
36. Steuer, R.: Multiple Criteria Optimization: Theory, Computation, and Application. Robert E. Krieger Publishing Company, Malabar (1989)
37. Valiente, G.: Algorithms on Trees and Graphs. Springer, Secaucus (2002)

38. Wu, B.Y., Chao, K.: Spanning Trees and Optimization Problems. Chapman & Hall/CRC, Boca Raton (2004)
39. Zhou, G., Gen, M.: Genetic algorithm approach on multi-criteria minimum spanning tree problem. Eur. J. Oper. Res. **114**(1), 141–152 (1999)