Pré-Publicações do Departamento de Matemática
Universidade de Coimbra
Preprint Number 17–53

A NEW APPROACH FOR THE MULTIOBJECTIVE MINIMUM SPANNING TREE

JOSÉ LUIS SANTOS, LUIGI DI PUGLIA PUGLIESE AND FRANCESCA GUERRIERO

Abstract: In this paper, a new algorithm for the multiobjective minimum spanning tree problem is presented that can be used with any number of criteria. It is based on a labelling algorithm for the multiobjective shortest path problem in a transformed network. Some restrictions are added to the paths (minimal paths) in order to obtain a one-to-one correspondence between trees in the original network and minimal paths in the transformed one. The correctness of the algorithm is proved as well as a short example is presented. Finally, some computational experiments were reported showing the proposed method outperforms the others existing in the literature. A deep study is also done about the number of nondominated solutions and a statistical model is presented to predict its variation in the number of nodes and criteria. All the test instances used are available through the web page http://www.mat.uc.pt/∼zeluis/INVESTIG/MOMST/momst.htm

Keywords: Multicriteria optimization, Minimum spanning tree.

1. Introduction

The minimum spanning tree (MST) problem is a classical combinatorial optimization problem studied since the beginning of last century. It was proposed by Borůvka in 1926 [4] and he describes an algorithm which runs in $O(m \ln n)$ time to solve a problem with n nodes and m edges. Other methods were proposed like the Prim's algorithm, which was first developed by Jarník [33] in 1930 and later rediscovered by Prim [46], in 1957, and Dijkstra [16], in 1959. It has time complexity of $O(m \ln n)$, but it can be improved to $O(m + n \ln n)$ if Fibonacci heap is used as data-structure [23]. Kruskal [37] proposed, in 1956, two methods: the well known Kruskal's algorithm which runs in $O(m \ln n)$ time and the reverse-delete algorithm with time complexity of $O(m \ln n(\ln \ln n)^3)$, using Thorup algorithm [55], working with the edges in the reverse cost order. A more detailed development of the beginnings of

Received December 19, 2017.

This work was partially supported by the Centre for Mathematics of the University of Coimbra – UID/MAT/00324/2013, funded by the Portuguese Government through FCT/MCTES and co-funded by the European Regional Development Fund through the Partnership Agreement PT2020.

the MST problem, its subsequent evolution and other methods can be found at [28, 30, 34, 42, 45].

The MST was motivated by a practical application in order to minimize the cost of the construction of a power line network in Southern Moravia after World War I [28]. Despite the practical motivation in the origin of the MST problem, some interesting theoretical properties can be derived from this problem [13] as well as other kind of applications in different fields of science, such as finances [5], ecotoxicology [14], clustering analysis [18], weather forecasting [26], image registration and segmentation [40]. The MST is also related with other combinatorial optimization problems like the Euclidian traveling salesman problem [9], the multi-terminal minimum cut problem [12], matroids [19], the minimum-cost weighted perfect matching [54] and the Steiner tree problem [57]. To obtain more details about MST properties, applications or related problems, the reader is referred to [3, 28, 58, 59, 61].

There are several variants of the MST problem in the literature like the Euclidean MST [1], the degree constrained MST [56], the k-smallest spanning trees [24], the arborescence [25], the hop-constrained and the diameter-constrained minimum spanning tree [27], the minimum labelling spanning tree [36], the maximum spanning tree [39], the capacitated MST [43], k-th MST [48], the delay MST [50] and the dynamic MST [52]. Other variants of the MST problem are presented in [58, 59].

In this work, we are focused on the multi-objective MST (MOMST) problem, which is a straightforward generalization of the classical MST problem where several weights are associated with each edge. In this problem, we aim to find the Pareto front, that is, a set of solutions for which it is not possible to improve one of the criteria without worsening any of the others. In what concerns to the complexity analysis, it is NP-complete [6] and intractable [31]. The first algorithm proposed to the MOMST problem is due to Corley in 1985 [11] which is based on Prim's algorithm, but it can lead us to a solution set with non-efficient spanning trees [31]. Hamacher [31] proved that the efficient set is not connected when two spanning tree are considered adjacent if they have $n-2$ edges in common. He also provides an exact algorithm to compute the supported efficient spanning trees in the bi-objective MST problem. The great part of the papers in the literature about this issue covers only the bi-objective case [10, 20]. Additionally, they deal mainly with approximate techniques, such as evolutionary algorithms [35, 41], genetic algorithm [32, 51, 60], local search [2] and ant colony [7, 38].

As far as we know, only 3 exact algorithms are proposed to find the full Pareto front: the exhaustive search, dynamic programming [15] and two-phase methods [47, 53]. The exhaustive search can only be used in a very small graphs because the number of spanning trees increases quickly with the number of the nodes; for instance, in a complete graph with n nodes, there are n^{n-2} spanning trees [8]. In the two-phase method, firstly it is computed the set of supported efficient spanning trees. Then, in the second phase, the unsupported efficient spanning trees are searched inside "viable regions" formed by two adjacent supported efficient solutions. Steiner and Radzik [53] showed that their approach using a ranking procedure to search the unsupported solutions outperforms the branch-and-bound technique applied in [47]. However, the two-phase method is designed for the bi-objective problems and the generalization for more than two criteria is hard to accomplish, [21]. An extended review of the existing methods and properties for the MOMST problem can be obtained in [7, 21, 22, 49].

In what concerns to the computational experiments, only one [15] of the papers analysed about the full computation of the Pareto front, reports results for more than two objectives. The remainder ones focus exclusively the bicriteria case. Furthermore, the lack of information about the tested problems prevents the replication of the computational experiments by other authors and the comparison of the results among different papers. In fact, in the papers analysed, the size of the networks varies from a dozen to hundreds of nodes, but there is no reference to the number of Pareto solutions. However, it is known the problem will be harder to solve the larger the number of solutions in the Pareto front. Depending on the type of network considered, it is possible to build a small network with a large number of nondominated solutions and a large network with the opposite situation.

The paper is organized as follows. In this section, an introduction to the MST and the MOMST is presented, while section 2 gives us the formal definition and notation used throughout the paper. Section 3 describes the algorithms which are used in section 4 to report our computational results. Finally, section 5 summarizes the paper and conclusions.

2. Definitions and notation

A graph (or an undirected graph) is an ordered pair $G = (N, E)$ where:

- $N = \{1, \ldots, n\}$ is the set of nodes;

- $E = \{e_1, \ldots, e_m\}$ is the set of edges, where each edge e_k ($k = 1, \ldots, m$) is a 2-element subset of N. Thus, e_k will be represented by an unordered pair $[i, j]$ with $i, j \in N$ and consequently $[i, j] = [j, i]$. Let E_i be the subset of E containing the edges linked to node i, that is, $E_i = \{e_\ell \in E : e_\ell = [i, j], \text{ for some } j \in N\}$. Multi-edges occur when there are different edges defined between the same pair of nodes.

A subgraph of G is a graph whose set of nodes is a subset of N and whose set of edges is a subset of E.

A path $p = \langle v_0, e_1, v_1, \ldots, e_\ell, v_\ell \rangle$ between s and t is a sequence of nodes and edges such that

- $v_i \in N$, $\forall i \in \{0, \ldots, \ell\}$;
- $s = v_0$ and $t = v_\ell$;
- $e_i = [v_{i-1}, v_i] \in E$, $\forall i \in \{1, \ldots, \ell\}$.

if $p = \langle v_0, e_1, v_1, \ldots, e_\ell, v_\ell \rangle$ and $q = \langle w_0, f_1, w_1, \ldots, f_k, w_k \rangle$ represent two paths in G where $v_\ell = w_0$, then $p \Diamond q = \langle v_0, e_1, v_1, \ldots, e_\ell, v_\ell = w_0, f_1, w_1, \ldots, f_k, w_k \rangle$ represents a path in G between v_0 and w_k.

A path can be represented only by the sequence of edges. If there are not multiple edges, it can also be represented only by the sequence of nodes. We denote by $\mathcal{P}_{s,t}$ the set of paths from node s to node t.

A graph G is connected if there is a path between each pair of nodes in N. A tree T is a connected subgraph of G with $n = |N|$ nodes and $n - 1$ edges. We denote by $\mathcal{T}_X$ the set of trees which set of nodes is $X \subseteq N$.

A graph is directed if each edge has an orientation. In this case, e_k will be represented by an ordered pair (i, j) and it will be called as *arc*. Note that in this case $(i, j) \neq (j, i)$. In a directed graph, multiple-arcs are defined as arcs between the same pair of nodes with the same direction. A path in a directed graph has to preserve the arc orientation. An undirected graph can be transformed into a directed graph $G^d = (N, A)$, where each edge $e = [i, j] \in E$ defines two arcs in A, (i, j) and (j, i), between i and j with opposite directions.

A network is a triplet (N, E, c) where:

- (N, E) is a graph;
- c is a vectorial cost function that assigns the vector cost $c([i, j]) \in \mathbb{R}^k$ to the edge $[i, j]$,

where k is the number of objectives or criteria. We will assume that all the components of the vector cost are nonnegatives.

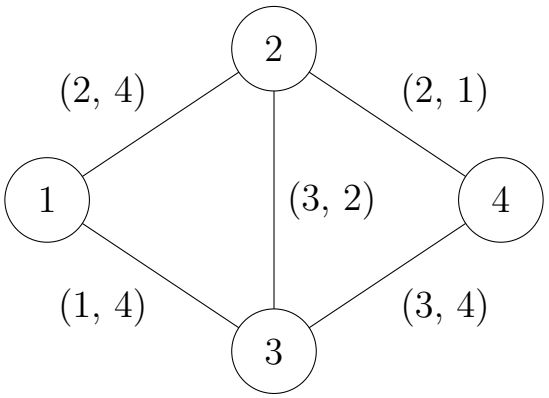

FIGURE 1. An undirected network $G = (N, E, c)$.

Figure 1 shows a network example that will be used throughout the paper. The cost of a path p, $c(p)$, is the sum of the cost for all edges in p. Then, $c(p) = \sum_{[i,j]\in p} c([i,j])$. Similarly, the cost of a tree T, $c(T)$, is defined by $c(T) = \sum_{[i,j]\in T} c([i,j])$. Similar concepts are defined for directed networks.

The multiobjective optimization is based on the concept of dominance which will be introduced next.

Definition 2.1. Let $\mathbf{a}$ and $\mathbf{b}$ be two vectors in $\mathbb{R}^k$. Then

$$\mathbf{a} \trianglelefteq \mathbf{b} \iff a_i \leq b_i,\ \forall i \in \{1,\ldots,k\} \text{ and } \mathbf{a} \neq \mathbf{b}.$$

In this case, we say $\mathbf{a}$ dominates $\mathbf{b}$ or $\mathbf{b}$ is dominated by $\mathbf{a}$.

Definition 2.2. Let $Y \subset \mathbb{R}^k$ and $\mathbf{a} \in Y$. $\mathbf{a}$ is a nondominated vector in Y if and only if $\nexists \mathbf{b} \in Y : \mathbf{b} \trianglelefteq \mathbf{a}$. We denote by Y^* the set of nondominated vectors in Y, that is, $Y^* = \{\mathbf{a} \in Y : \nexists \mathbf{b} \in Y : \mathbf{b} \trianglelefteq \mathbf{a}\}$.

Definition 2.3. Let s and t be two nodes of N. A path $p \in \mathcal{P}_{s,t}$ is said to be efficient if and only if $c(p) \in c(\mathcal{P}_{s,t})^*$, that is, $c(p)$ is a nondominated vector in $c(\mathcal{P}_{s,t})$.

Definition 2.4. Let X be a subset of N. A tree $T \in \mathcal{T}_X$ is said to be efficient if and only if $c(T) \in c(\mathcal{T}_X)^*$, that is, $c(T)$ is a nondominated vector in $c(\mathcal{T}_X)$.

The Pareto optimal solution of the MOMST is a subset $\mathcal{T}^* \in \mathcal{T}_N$ for which $c(\mathcal{T}^*) = c(\mathcal{T}_N)^*$. The set $c(\mathcal{T}_N)^*$ is also called the Pareto front.

3. The new algorithm

3.1. Built network. Given an undirected network $G = (N, E, c)$, the built network $G^{\mathcal{T}} = (N^{\mathcal{T}}, A^{\mathcal{T}}, c^{\mathcal{T}})$ is a directed network (with multiple-arcs) defined in the following way:

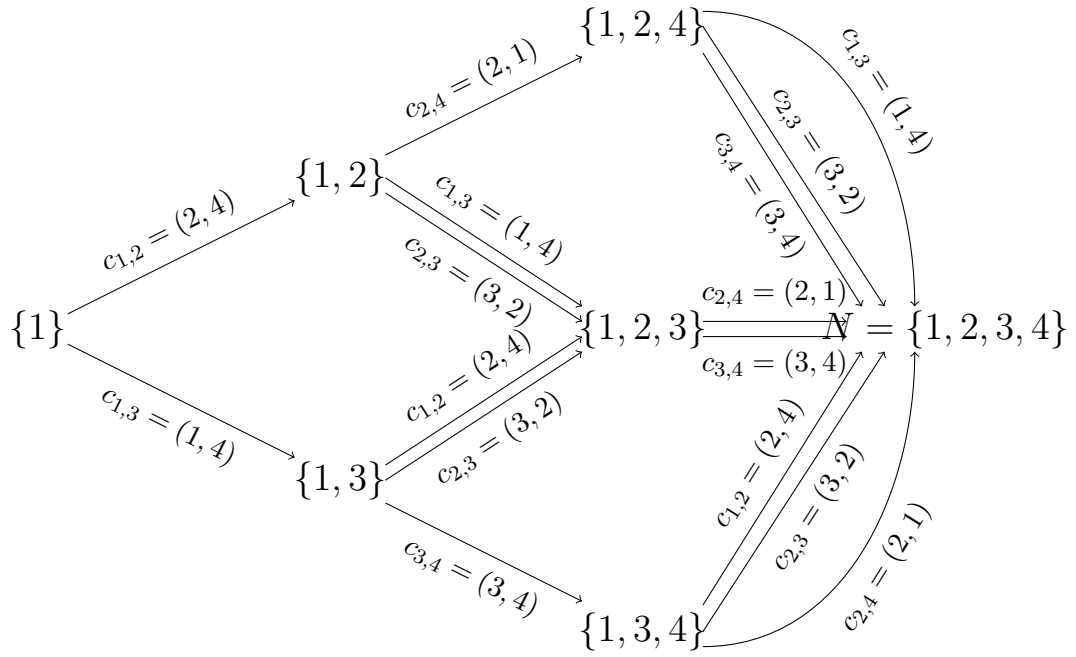

FIGURE 2. The BN network $G^{\mathcal{T}}$ obtained from G in Figure 1.

- $N^{\mathcal{T}} = \{\{1\} \cup X : X \subseteq N \backslash \{1\}\}$ is the set of nodes ($|N^{\mathcal{T}}| = 2^{n-1}$);
- $A^{\mathcal{T}} = \{(X, X \cup \{j\})_{[i,j]} : X \in N^{\mathcal{T}}, i \in X, j \notin X, [i,j] \in E\}$ is the set of arcs;
- $c^{\mathcal{T}}$ is the vectorial cost function, where $c^{\mathcal{T}}((X, X \cup \{j\})_{[i,j]}) = c([i,j])$.

Note that $G^{\mathcal{T}}$ is a layered network, where layer i contains the nodes of $N^{\mathcal{T}}$ with $i+1$ elements (that is, with $i+1$ nodes of N) and the arcs link nodes between two consecutive layers. Furthermore, the first and the last layers are formed by a single node of $N^{\mathcal{T}}$ which are $\{1\}$ and N, respectively. Figure 2 shows the network $G^{\mathcal{T}}$ obtained from G in Figure 1. We would like to emphasize that $G^{\mathcal{T}}$ is not explicitly built in the algorithm, as it will be discussed later. This network is described in the paper to understand how paths in $G^{\mathcal{T}}$ are built.

In order to recover the information of the initial network G from $G^{\mathcal{T}}$, it is defined the function $edge : A^{\mathcal{T}} \to E$ such that $edge(\,(X, X \cup \{j\})_{[i,j]}\,) = [i,j]$.

The following properties are the basis of this model and establish the relation between trees in G and paths in $G^{\mathcal{T}}$.

3.2. Relations between $G^{\mathcal{T}}$ and G.

Proposition 3.1. *Let $p = \langle X_0, a_1, X_1, \ldots, a_{n-1}, X_{n-1}\rangle$ be a path in $G^{\mathcal{T}}$. Then, $T^p = \{edge(a_1), \ldots, edge(a_{n-1})\}$ is a tree in G. Additionally, T^p is the unique tree of G represented by p.*

Proof: We use induction to prove that $T^{p_k} = (X_k, \cup_{\ell=1}^{k}\{edge(a_\ell)\})$,$1 \leq k < n$, is a connected sub-graph with $k+1$ nodes and k edges; that is, T^{p_k} is a sub-tree in G.
Let $p_1 = \langle X_0, a_1, X_1\rangle$ be a path in $G^{\mathcal{T}}$. Then, $X_0 = \{1\}$, $X_1 = X_0 \cup \{j_1\}$, $a_1 = (X_0, X_1)_{[i_1,j_1]}$ with $i_1 = 1$, $j_1 \neq 1$ and $[i_1, j_1] \in E$. Additionally, $X_1 = \{1, j_1\}$ has two different nodes and consequently T^{p_1} is a sub-tree in G.
Suppose now that $p_{k+1} = \langle X_0, a_1, X_1, \ldots, X_k, a_{k+1}, X_{k+1}\rangle$, $1 \leq k < n-1$, is a sub-path in $G^{\mathcal{T}}$ and the corresponding sub-graph T^{p_k} is a sub-tree in G. We will prove that $T^{p_{k+1}}$ is also a sub-tree in G. In fact, by definition of p_{k+1}, $X_{k+1} = X_k \cup \{j_{k+1}\}$, $a_{k+1} = (X_k, X_{k+1})_{[i_{k+1},j_{k+1}]}$ with $i_{k+1} \in X_k$, $j_{k+1} \notin X_k$ and $[i_{k+1}, j_{k+1}] \in E$. Then, the sub-graph $T^{p_{k+1}}$ is a connected sub-graph of G with $k+1$ nodes of N, and k edges of E; that is, $T^{p_{k+1}}$ is a sub-tree in G. Finally, the set $\cup_{\ell=1}^{k+1}\{edge(a_\ell)\}$, with $k+1$ arcs, defines a unique tree in G.

Proposition 3.2. *Let $p = \langle X_0, a_1, X_1, \ldots, a_{n-1}, X_{n-1}\rangle$ be a path in $G^{\mathcal{T}}$ and T^p the corresponding tree in G. Then, $c^{\mathcal{T}}(p) = \mathbf{c}(T^p)$.*

Proof: It follows directly by the definition of the path p and the tree T^p.

In order to construct a path p^T in $G^{\mathcal{T}}$ that corresponds to a tree T in G the Algorithm 1 is defined.

Proposition 3.3. *Given a tree T in G, Algorithm 1 defines a path in $G^{\mathcal{T}}$ from $\{1\}$ to N.*

Proof: The algorithm iterates $n-1$ times because Z starts with $n-1$ edges in line 1 and, at each iteration of cycle in lines 3-9, it decreases one unit. Then, the stop condition in line 3 is achieved and the cycle finishes. Moreover, the edge in line 4 exists because T is a tree. Line 6 allows to extend p with a new arc and a new node following the rules established in section 3.1, so p is updated with a new path in $G^{\mathcal{T}}$. Next, the edge $[i, j]$ is removed from Z while X is updated with node j, containing all the nodes already visited by the algorithm. Consequently, $X = N$ at the end of the

Input: a tree T in G
Output: a path p^T in $G^{\mathcal{T}}$ such that $T^{p^T} = T$

```
// Main variables:
//    Z:  set of unscanned edges in T
//    X:  set of nodes already visited
```

```
Z ← set of edges in T;
X ← {1};   p ← ⟨{1}⟩;
while Z ≠ ∅ do
    [i, j] ← an edge of {[i, j] ∈ Z : i ∈ X, j ∉ X};
    X' ← X ∪ {j};
    p ← p◊⟨X, (X, X')_[i,j], X'⟩ ;
    Z ← Z\{[i, j]};
    X ← X';
end
p^T ← p;
```

Algorithm 1: Construction of a path p^T in $G^{\mathcal{T}}$ such that $T^{p^T} = T$.

algorithm and p is a path from $\{1\}$ to N.

Proposition 3.4. *Given a tree T in G, a path p^T produced by Algorithm 1 verifies $T^{p^T} = T$ and $c(p^T) = c(T)$.*

Proof: It follows by the way as p^T was constructed.

3.3. Minimal path. The previous proposition establishes that searching all the paths in $G^{\mathcal{T}}$ corresponds to search all trees in G. However, there are several paths p in $G^{\mathcal{T}}$ that satisfies $T^{p^T} = T$ and, consequently, we must avoid searching several times the same tree in G. To this purpose, the following definition is introduced.

Definition 3.5. Let $p = \langle X_0, a_1, X_1, \ldots, a_k, X_k \rangle$ be a path in $G^{\mathcal{T}}$ and suppose that $edge(a_\ell) = [i_\ell, j_\ell]$, $1 \leq \ell \leq k$. p is said to be a "*minimal path*" if it satisfies one of the following conditions:

- $k = 1$

- $k \geq 2$, $\langle X_0, a_1, X_1, \ldots, a_{k-1}, X_{k-1}\rangle$ is a minimal path and one of the following additional conditions is held
  - $i_{k-1} = i_k$ and $j_{k-1} < j_k$,
  - $\exists \ell, \ell' \in \{1, \ldots, k\} : \ell < \ell', i_{k-1} = j_\ell$ and $i_k = j_{\ell'}$.

This definition states that every path in $G^{\mathcal{T}}$ with a single arc is a minimal path. Furthermore, suppose that a minimal path p in $G^{\mathcal{T}}$ was lastly extended using an edge $e_k = [i, j]$ of node i. Then, there are two ways to obtain minimal paths from p. The first one is using edges $[i, j']$ of the same node i but with $j' > j$. Alternatively, p can be extended with edges from a node included in p after node i (the index ℓ in j_ℓ represents the order in which nodes are entering into the path).

Algorithm 2 allows us to construct a minimal path p^T in $G^{\mathcal{T}}$ that corresponds to a tree T in G. The main difference between algorithms 1 and 2 is the way how they select the next edge to be included in the path. Consequently, proposition 3.3 and 3.4 are also valid for the path produced by Algorithm 2. In lines 7 and 9 it is defined (in a unique way) the next edge to expand p. Consequently, given a tree T in G, only one path can be produced by Algorithm 2 and the following proposition states that it is minimal.

Proposition 3.6. *Given a tree T in G, Algorithm 2 defines a (unique) minimal path in $G^{\mathcal{T}}$ from $\{1\}$ to N.*

Proof: Likewise Algorithm 1, Algorithm 2 ends after $n-1$ iterations with a path in $G^{\mathcal{T}}$ from $\{1\}$ to N. To prove it is minimal, induction will be used. In fact, at the first iteration ($k = 1$), the path p obtained in line 11 is minimal because it has only one arc. Suppose now that, at iteration $k < n - 1$, p is still a minimal path. Note that, the last arc added to p was obtained from an edge of "*pivot*", that is, $i_k = pivot$. If the condition in line 6 is false, then there are more unscanned edges of T from "*pivot*" and "*pivot*" will be not updated, that is $i_{k+1} = pivot$. Thus, the edge $[pivot, j]$ chosen in line 9 is the next (minimum) unscanned edges of T from "*pivot*" and, as a result, $[i_{k+1}, j_{k+1}] = [pivot, j]$ with $j > j_k$ (otherwise, j_k will be not the minimum at line 9 in the previous iteration). Finally, if $Z \cap E_{pivot} = \emptyset$ then line 7 updates "*pivot*" with the following node included in p with available edges in Z. This node exists because T is a tree and nodes are searched following *ord* value. Therefore, the new "*pivot*" (i.e., i_{k+1}) holds $ord(i_{k+1}) > ord(i_k)$. In both cases, p is updated with a minimal path on line 11.

Input: a tree T in G
Output: a minimal path p^T in $G^{\mathcal{T}}$ such that $T^{p^T} = T$

// Main variables:
// Z: set of unscanned edges in T
// X: set of nodes already visited
// $ord(k)$: the k-th node included in the path

$Z \leftarrow$ set of edges in T;
$X \leftarrow \{1\}$; $p \leftarrow \langle\{1\}\rangle$;
$k \leftarrow 1$; $ord(1) \leftarrow 1$;
$pivot \leftarrow 1$;
while $Z \neq \emptyset$ **do**
  if $Z \cap E_{pivot} = \emptyset$ **then**
    $pivot \leftarrow$
    $argmin_{v \in X}\{ord(v) : ord(v) > ord(pivot) \ \wedge \ Z \cap E_v \neq \emptyset\}$;
  end
  $j \leftarrow min\{j \in N : [pivot, j] \in Z\}$;
  $X' \leftarrow X \cup \{j\}$;
  $p \leftarrow p \Diamond \langle X, (X, X')_{[i,j]}, X' \rangle$;
  $Z \leftarrow Z \backslash \{[i, j]\}$; $X \leftarrow X'$;
  $k \leftarrow k + 1$;
  $ord(j) \leftarrow k$;
end
$p^T \leftarrow p$;

Algorithm 2: Construction of the minimal path p^T, associated to the tree T, such that $T^{p^T} = T$.

Proposition 3.7. *There is a one to one correspondence between trees in G and minimal paths in $G^{\mathcal{T}}$.*

Proof: It follows directly from propositions 3.1 and 3.6.

Finally, Algorithm 3 allows us to search all the minimal paths in $G^{\mathcal{T}}$ from $\{1\}$ to N and, consequently, the entire set of spanning trees in G. It uses two functions "$sequenceNodes$" and "$searchNewMinimalPaths$" which are

Input: a graph $G = (N, E)$
Output: set with all minimal paths in $G^{\mathcal{T}}$ from $\{1\}$ to N

```
// Main variables:
//    U: set of unscanned (minimal) subpaths in G^T
//    S(X): set of minimal paths associated X ⊆ N
//    p: current path under analysis
//    f(p): path from which p was obtained
//    e(p): last edge of G added to path p
```

$p \leftarrow \langle\{1\}\rangle$; $X \leftarrow \{1\}$;
$f(p) \leftarrow$ NULL; $e(p) \leftarrow [1, 1]$;
$U \leftarrow \{p\}$; $S(X) \leftarrow \{p\}$;
while $U \neq \emptyset$ **do**
  $p \leftarrow$ a path from U;
  $\langle u_1, \ldots, u_k\rangle \leftarrow$ sequenceNodes(p);
  $X \leftarrow \{u_1, \ldots, u_k\}$;
  $[i, j] \leftarrow e(p)$;
  $\ell \leftarrow$ index of i in $\langle u_1, \ldots, u_k\rangle$;
  searchNewMinimalPaths(p, i, j, X);
  for $x \leftarrow \ell + 1$ **to** k **do**
    searchNewMinimalPaths($p, u_x, 0, X$);
  end
end
$U \leftarrow U \backslash \{p\}$;
return $S(N)$;

Algorithm 3: Searching all minimal paths in $G^{\mathcal{T}}$.

defined in Algorithm 4. The validation of the algorithm is a consequence of proposition 3.7.

The algorithm is applied to the network example (Figure 1) to better understand the way it works and Figure 3 shows all the minimal paths in $G^{\mathcal{T}}$. In this figure, each branch corresponds to a minimal subpath and the arcs are labelled with the corresponding edge of the original network. Additionally, the nodes belonging to each set X appear in the order they were introduced into the minimal path and each branch is labelled with the cost of the corresponding minimal subpath. The algorithm starts with

```
function sequenceNodes(p);
begin
    [i, j] ← e(p);
    if f(p) == NULL then
        return ⟨j⟩;
    else
        return sequenceNodes(f(p)) ◊ ⟨j⟩;
end
```

```
function searchNewMinimalPaths(p, i, j, X);
begin
    for w ← j + 1 to |N| do
        if [i, w] ∈ E and w ∉ X then
            X′ ← X ∪ {w};
            p′ ← p◊⟨X, (X, X′)_[i,w], X′⟩;
            f(p′) ← p;   e(p′) ← [i, w];
            U ← U ∪ {p′};   S(X′) ← S(X′) ∪ {p′};
        end
    end
end
```

Algorithm 4: Functions used in algorithm 3.

$p_1 = \langle\{1\}\rangle$ which corresponds to the root of the search tree. From p_1, two minimal subpaths are created: $p_2 = \langle\{1\}, (\{1\}, \{1,2\})_{[1,2]}, \{\underline{1}, 2\}\rangle$ and $p_3 = \langle\{1\}, (\{1\}, \{1,3\})_{[1,3]}, \{\underline{1}, 3\}\rangle$ using, respectively, edges $[1,2]$ and $[1,3]$. Node 1 appears underlined because it corresponds to the *pivot* node in Algorithm 2 and means the last edge added to these paths belong to E_1. In this example, the FIFO (First In First Out) rule is used to choose the next path p to be expanded (line 5 of Algorithm 3). Following this rule, p_2 is chosen and $[1,2]$ is the last edge used to obtain p_2. From p_2 three new minimal subpaths are obtained towards the edges $[1,3]$, $[2,3]$ and $[2,4]$, respectively. The next path chosen in line 5 following the FIFO rule is p_3 which can be expanded through the edges $[3,2]$ and $[3,4]$. Note that edge $[1,2]$ is unconsidered because the subpath $q = \langle\{1\}, (\{1\}, \{1,3\})_{[1,3]}, \{\underline{1}, 3\}, (\{1,3\}, \{1,3,2\})_{[1,2]}, \{\underline{1}, 3, 2\}\rangle$ is not minimal. This procedure is repeated until there are no more minimal subpaths to expand.

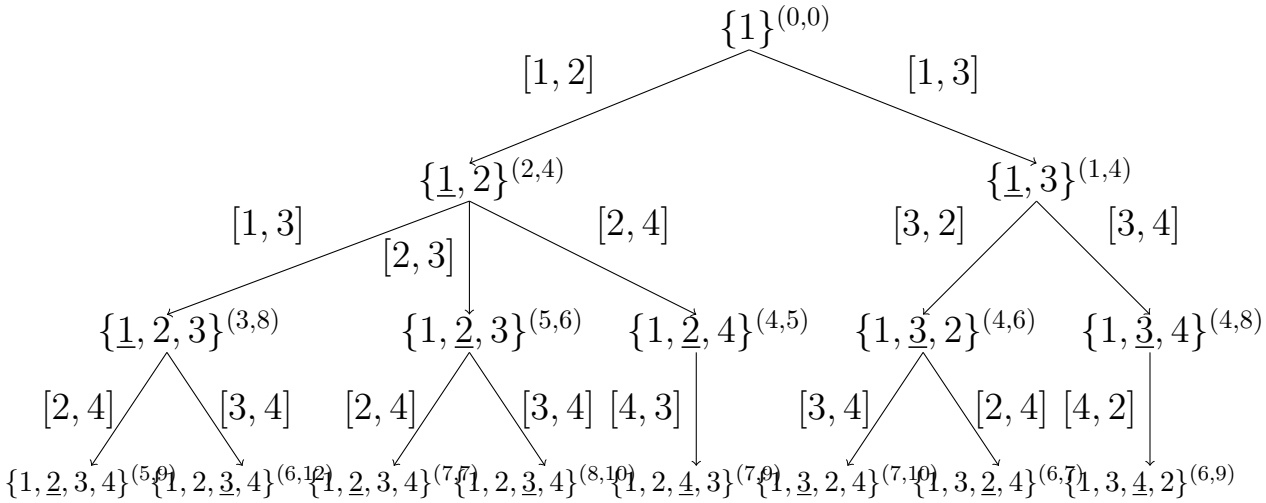

FIGURE 3. The search tree for all minimal paths in $G^{\mathcal{T}}$ for the example of Figure 1.

4. The BN algorithm

In the previous example, several minimal subpaths with the same terminal node were obtained. Consequently, the dominance test can be applied on them. As an example, when subpath

$$p_1 = \langle \{1\}, (\{1\}, \{1,3\})_{[1,3]}, \{\underline{1},3\}, (\{1,3\}, \{1,3,2\})_{[3,2]}, \{1,\underline{3},2\} \rangle$$

is generated from $\langle \{1\}, (\{1\}, \{1,3\})_{[1,3]}, \{\underline{1},3\} \rangle$, it should be compared with the others subpaths from $\{1\}$ to $\{1,2,3\}$ already computed, that is, with subpaths

$$p_2 = \langle \{1\}, (\{1\}, \{1,2\})_{[1,2]}, \{\underline{1},2\}, (\{1,2\}, \{1,2,3\})_{[1,3]}, \{\underline{1},2,3\} \rangle$$

and

$$p_3 = \langle \{1\}, (\{1\}, \{1,2\})_{[1,2]}, \{\underline{1},2\}, (\{1,2\}, \{1,2,3\})_{[2,3]}, \{1,\underline{2},3\} \rangle.$$

As p_3 dominates p_1, p_1 will be discarded. Following this idea, the proposed algorithm is based on Algorithm 3 by adding the dominance test in the search of new minimal paths. In this way, "searchNewMinimalPaths" routine is changed by "searchNewNDminimalPaths" which is sketched in Algorithm 5. Consequently, the BN algorithm can be viewed as a labelling algorithm for the multiobjective shortest minimal path problem in network $G^{\mathcal{T}}$. In the previous section, the FIFO rule was used to select the next minimal path to be expanded and then the BN algorithm can be viewed as a label correcting version. However, others rules can be used at this step obtaining different versions of the labelling algorithm. In this paper, the FIFO rule was selected

```
1  function searchNewNDminimalPaths(p, i, j, Y);
2  begin
3      for w ← j + 1 to |N| do
4          if [i, w] ∈ E and w ∉ Y then
5              Y' ← Y ∪ {w};
6              p' ← p◊⟨Y, (Y, Y')_[i,w], Y'⟩;
7              if p' is not dominated by paths in S(Y') then
8                  f(p') ← p;   e(p') ← [i, w];
9                  U ← U ∪ {p'};
10                 S(Y') ←
                   (S(Y') ∪ {p'})\{paths of S(Y') dominated by p'};
11             end
12         end
13     end
14 end
```

Algorithm 5: Function "searchNewNDminimalPaths" used in the BN algorithm.

because it leads to one of the most efficient labelling approaches for the multiobjective shortest path problem [29, 44].

The validation of the algorithm is based on the following facts:

- Algorithm 3 searches all minimal paths in $G^{\mathcal{T}}$,
- there is a one to one correspondence between minimal paths in $G^{\mathcal{T}}$ from $\{1\}$ to N and spanning trees in G,
- the optimality principle is held for multiobjective shortest path problem.

Figure 4 shows the final search tree produced by the BN algorithm in the network example pictured in Figure 1. In this figure, gray color is used to represent dominated minimal subpaths. Moreover, dashed lines correspond to dominated branches that are promptly eliminated in line 7. Therefore, these subpaths will never be stored by algorithm 5. On the other hand, solid lines correspond to branches that were not dominated when the dominance test is performed in line 7. Note that some of these branch can turn into dominated ones when more subpaths are generated and the dominance test is performed in line 10. This is the case of subpath $\langle\{1\}, \{\underline{1}, 2\}, \{1, \underline{2}, 3\}\rangle$, which

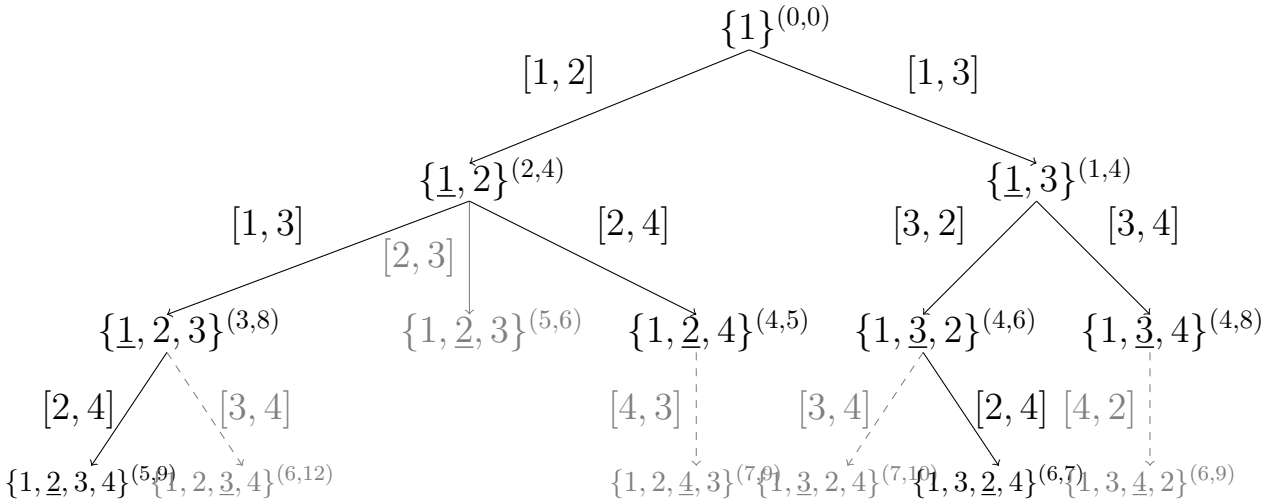

FIGURE 4. Final search tree produced by the BN algorithm in the network example pictured in Figure 1.

was not dominated by $\langle\{1\},\{\underline{1},2\},\{\underline{1},2,3\}\rangle$ but it becomes a dominated subpath when it is compared with $\langle\{1\},\{\underline{1},3\},\{1,\underline{3},2\}\rangle$. Note that the BN algorithm works directly with the edges of the initial network G and the built network $G^{\mathcal{T}}$ was never explicitly used.

5. Computational results

In this section we compare the performance of the new algorithm (BN) with the dynamic programming (DP) approach and the TP Steiner Radzik (SR) approach. All the codes were written in JAVA and they were run on a Intel(R) core(TM) i7-4720HQ CPU 2.60GHz 8GB RAM machine. We used the *spacyc* generator [35] to obtain a set of test instances which includes instances with 2, 3 and 4 criteria (k). For each number of criteria, we consider instances with 5 up to 14 nodes (n) with $m = n \times d$ edges, where $d \in \{5, 10, 15, 20\}$. The costs associated with each arc are randomly generated by using the generator of [35]. They have been chosen in the interval [0,100] and the criteria are not correlated.

The computational experiments were conducted in two stages. At the first stage, the three algorithms were compared only on bicriteria instances. In order to be more accurate, 80 instances of the same type were considered, that is 80 instances with different "seeds" but with the same value of k, n and d. The reported values correspond to the average results regardless the value of d and "seed". In the second stage, only BN and DP were

$k\backslash n$	5	6	7	8	9	10	11	12	13	14
2	7.05	10.90	16.44	22.63	24.02	34.23	46.56	48.74	57.67	67.91
3	16.65	42.90	92.10	135.15	256.80	458.20	549.10	910.05	1206.20	1909.60
4	29.30	118.60	269.65	661.60	1370.95	3629.15	5426.50	10985.80	14881.60	24679.75

TABLE 1. Average number of nondominated solution in terms of n and k

compared on instances with 3 and 4 criteria. The SR approach was not considered at this stage because we only implemented the bicriterion version of this algorithm. As these instances are much harder to solve, only 20 instances of each type were considered. Table 1 reports the average number of nondominated spanning trees in terms of n and k. All the instances are available in

http://www.mat.uc.pt/∼zeluis/INVESTIG/MOMST/momst.htm

to make these results reproducible in other works.

The algorithms were compared in terms of CPU time (in milliseconds) to evaluate its performance. This section also includes a detailed analysis of the number of Pareto optimal solutions.

5.1. Detailed analysis on the number of Pareto optimal solutions. In this subsection we propose the following model

$$PO = ae^{bnk}, \quad a, b \in \mathbb{R}^+ \tag{5.1}$$

to predict the variation of the number of Pareto Optimal solutions (PO) with the network parameters n and k. Equation (5.1) was log transformed in order to obtain the following linear model with the additional white noise term:

$$Y = a' + b'x + \varepsilon, \quad a', b' \in \mathbb{R} \tag{5.2}$$

where $Y = \ln PO$, $x = nk$, $a = e^{a'}$, $b = b'$ and $\varepsilon \sim N(0, \sigma)$ with $\sigma > 0$ the standard deviation of ε.

Parameters a', b' and ε were obtained by linear regression [17] over the set of 1200 instances solved. Table 2 reports a summary of the statistic analysis for the model presented in equation (5.2) and Figure 5 shows the 95% confidence bands for the individual value of Y. The Pearson correlation coefficient obtained in the experiments was $R = 0.952527$ revealing a strong correlation between $\ln(PO)$ and the product nk. From this value, the coefficient of determination $R^2 = 0.907307$ is computed and it indicates that more than 90% of the results are explained by the linear model.

Parameter	Estimative	Confidence interval	
a'	-0.074555	]-0.160149, 0.011038[	
b'	0.182097	]0.178798, 0.185396[	$R^2 = 0.907307$
σ	0.608226	]0.584819, 0.633599[	
$Y(x)$	$a' + b'x$	$]a' + b'x - g(x), a' + b'x + g(x)[$	$R = 0.952527$

$$g(x) = 0.003299\sqrt{130934.020833 + (x - 23.75)^2}$$

TABLE 2. Summary of the statistic analysis for the model presented in equation (5.2) with confidence level = 0.95.

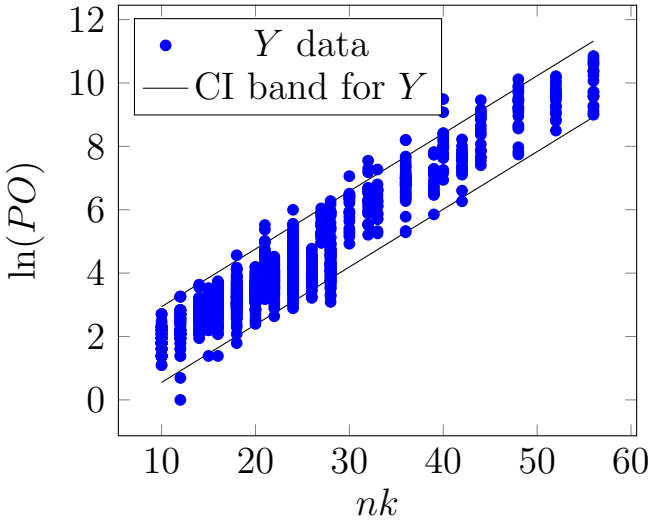

FIGURE 5. Linear regression model and confidence intervals for $\ln(PO)$.

To validate this model, it was also tested if the error $\varepsilon = Y - (a' + b'x)$ is normally distributed. Figure 6 shows the histogram obtained with experimental data and the corresponding fitted normal distribution (left) as well as the Q-Q plot (right). From this figures, normal distribution can be assumed for ε. Table 3 reports the descriptives and tests for normality of ε which also attest the assumption made.

Taking into account these results, it is possible to obtain the following model for the number of Pareto optimal solutions in terms of n and k:

$$PO = 0.928156e^{0.182097nk}. \tag{5.3}$$

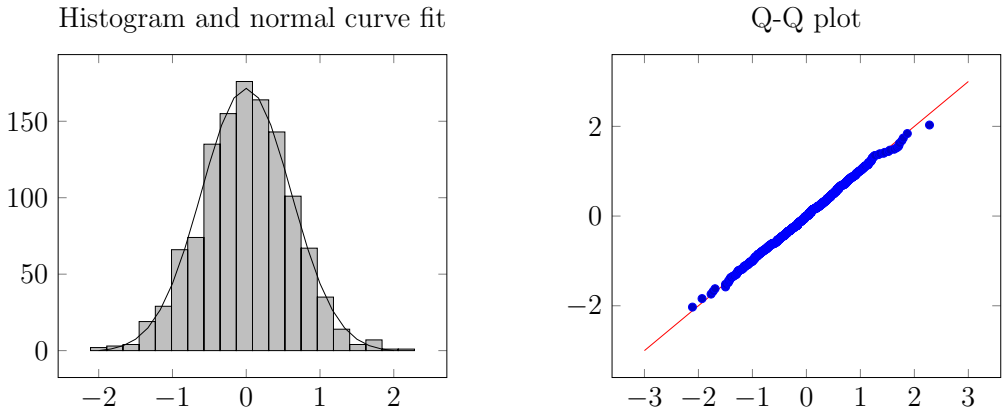

FIGURE 6. Histogram for data error and respective Q-Q plot graph.

Descriptives

Min:	-2.11	Median:	0.0012	Standard deviation:		Skewness:	-0.025
Max:	2.28	Mean:	0.0000	0.60797		Kurtosis:	0.142

p-value for normality tests

Kolmogorov-Smirnov (with a Lilliefors significance correction)	Shapiro-Wilk
0.093	0.592

TABLE 3. Descriptives and tests for normality of ε.

With this model, it is possible to estimate the number of PO solutions in terms of n and k. Figure 7 shows the estimative for the number of PO solutions with formula (5.3) which are in agreement with the average values obtained in our computational tests.

5.2. Computational results - first stage ($k = 2$). In this subsection we analyze the performance, in terms of CPU time (in milliseconds), of the BN, DP and SR algorithms on bicriteria instances. Table 4 reports the average, standard deviation and the maximum CPU time needed to solve the instances for each number of nodes considered in our experiments and Figure 8 depicted (in log scale) the average CPU time for clear visualization. From these results, it is concluded that CPU time seems to increase exponentially with the number of nodes in all the algorithms. Similar behaviour is observed for the standard deviation and the maximum CPU time. Additionally, in our tests, BN is the algorithm with better performance in terms of average,

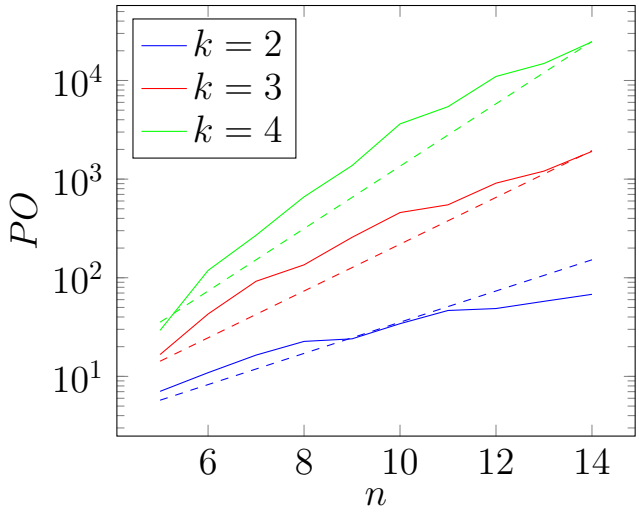

FIGURE 7. Predicted value for the number of PO solutions and the average value in the computational tests (solid lines: average value in the computational tests; dashed lines: estimative based in (5.3)).

	Average			Standard deviation			Maximum		
n	DP	SR	BN	DP	SR	BN	DP	SR	BN
5	0.51	0.60	0.18	2.49	2.93	1.46	16	16	13
6	2.15	1.03	0.20	6.29	3.32	1.68	47	16	15
7	9.27	3.01	0.61	9.71	6.49	3.06	41	31	16
8	81.42	12.88	1.58	77.99	16.43	4.75	649	99	16
9	608.92	14.94	3.26	647.34	34.41	6.15	4160	285	18
10	6591.51	77.39	17.54	5285.62	122.50	14.61	25592	623	110
11	37816.66	292.09	41.08	26954.18	543.62	20.69	130453	2907	87
12	230931.33	314.07	110.91	207009.07	657.57	53.91	969201	4076	219
13	1248728.65	538.99	344.52	1091665.37	947.11	195.14	5670391	6118	1063
14	7994751.35	768.98	1122.22	5380345.28	798.52	588.21	21427829	5294	3349

TABLE 4. Average, standard deviation and the maximum CPU time in bicriteria instances

standard deviation and maximum value while DP is the worst. The unique exception occurs for $n = 14$, where SR performs slightly better than BN in terms of average CPU time. Consequently, we can conclude that, in general, BN is the fastest algorithm in our instances and has the smallest variability in the CPU time for instances with the same characteristics (see Standard deviation).

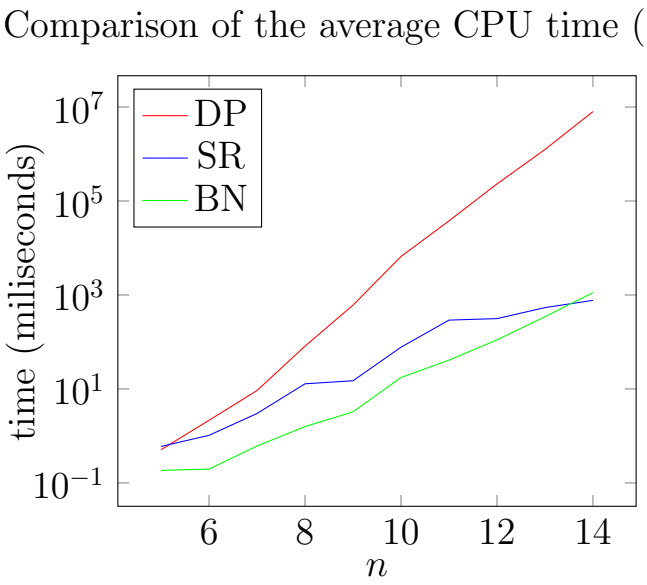

FIGURE 8. Average CPU time with the number of nodes on bicriteria instances.

In order to support our conclusion, Table 5 presents the one-tailed p-value, using pairwise data, for testing if the average values of CPU time applying algorithm A is significantly smaller than the ones obtained with algorithm B, where the pair $A - B$ is reported in the first column. From this table, for $n > 6$, it is observed that p-value is, in general, less than 1% which allows supporting our conclusion about the performance of the algorithms.

n	5	6	7	8	9	10	11	12	13	14
BN-DP	0.2348	0.0128	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
BN-SR	0.2127	0.0575	0.0060	0.0000	0.0054	0.0001	0.0001	0.0078	0.0555	0.9999
SR-DP	0.6108	0.0964	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

TABLE 5. One-tailed p-value of the t-test for the difference of mean in pairwise data (the value 0.0000 means that p-value $<$ 0.00005).

5.3. Computational results - second stage ($k \in \{3, 4\}$). As already mentioned, in this subsection only BN and DP were tested because our implementation of SR only runs on bicriteria instances. Table 6 reports the average, standard deviation and maximum CPU time in both algorithms. As in the previous case, Figure 9 shows the average CPU time (in log scale), where it is noticed a similar performance than in bicriteria instances. The

	$k = 3$					
	Average		Standard deviation		Maximum	
n	DP	BN	DP	BN	DP	BN
5	1.66	0.16	4.75	0.36	16	1
6	6.70	0.21	8.01	0.41	24	1
7	67.80	2.36	70.61	5.74	336	16
8	707.45	7.60	428.94	8.28	1696	22
9	10029.20	33.45	5030.96	12.00	20992	56
10	132211.95	253.70	134545.36	235.29	586341	966
11	1127480.35	1000.45	1304066.85	1016.87	4863076	3940
12	14235842.85	5424.70	12731813.06	4693.19	49204034	19076
13	46857124.67	22113.85	31928411.42	21130.07	83669553	91662
14	d.n.r.	86331.35	d.n.r.	59193.35	d.n.r.	208307

	$k = 4$					
	Average		Standard deviation		Maximum	
n	DP	BN	DP	BN	DP	BN
5	2.80	0.21	5.58	0.41	16	1
6	16.65	0.36	19.28	0.58	88	2
7	231.30	6.80	160.10	8.62	631	30
8	6402.85	58.70	4312.35	52.25	16732	207
9	77419.40	452.45	75381.79	513.38	323085	2177
10	2492525.10	5584.60	2938240.59	8499.67	13036565	39159
11	25706044.40	28404.45	21495708.56	26262.08	64897090	72563
12	d.n.r.	188144.15	d.n.r.	172464.39	d.n.r.	565968
13	d.n.r.	712957.60	d.n.r.	701483.45	d.n.r.	3195679
14	d.n.r.	3656910.70	d.n.r.	4035560.88	d.n.r.	15600000

TABLE 6. Average, standard deviation and the maximum CPU time on instances with 3 and 4 criteria (d.n.r.: did not run).

same results were observed for the standard deviation and maximum CPU time in both algorithms. From these results, it is concluded BN algorithm obtains the best performance in terms of the average CPU time. It is noteworthy that BN algorithm has less variability in terms of CPU time for instances with the same characteristics (number of nodes and criteria). Additionally, as it was expected, problems with more criteria demand higher computational effort.

6. Conclusion

In this paper, a new algorithm for the MOMST problem is presented that works with any number of criteria. The key idea is to transform the original

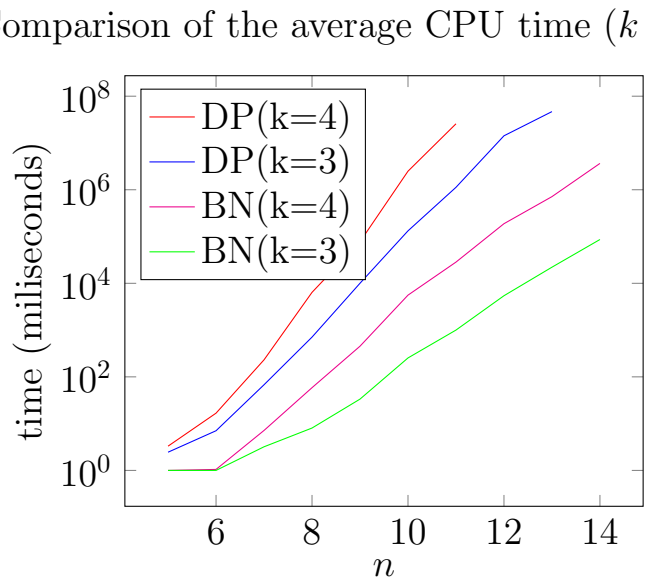

FIGURE 9. Average CPU time with the number of nodes on instances with 3 and 4 criteria.

network in a directed network where trees in the first one correspond to paths in the second one. Some restrictions are made on the paths (minimal paths) in order to obtain a one-to-one correspondence between both sets of solutions. The restrictions can be easily implemented in a labelling algorithm. In the example and the reported computational results, the FIFO rule was used to choose the next minimal path to be expanded. In this way, the proposed algorithm is based on the label correcting algorithm for the multiobjective shortest path problem. However, other rules can be used allowing to use, for instance, the label setting algorithm. A small example is given in order to show how the algorithm works.

Some properties of the transformed network were discussed which allows us to prove the correctness of the algorithm. The new algorithm was compared with the dynamic programming and two-phase approaches over the set of 1200 instances with up to 4 criteria. The computational results show the superiority of the new method. A web page

http://www.mat.uc.pt/∼zeluis/INVESTIG/MOMST/momst.htm

was developed containing all the test instances used in this paper to be available in future works.

The computational effort needed to solve the MOMST is strictly dependent on the number of non-dominated solutions. Thus, a statistical model was presented to predict it in terms of the number of nodes and criteria in the

tested instances. From this model, the number of non-dominated minimum spanning tree grows exponentially with the product of the number of nodes and criteria.

References

[1] P. K. Agarwal, H. Edelsbrunner, O. Schwarzkopf, and E. Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. *Discrete & Computational Geometry*, 6(3):407–422, 1991.

[2] K. A. Andersen, K. Jörnsten, and M. Lind. On bicriterion minimal spanning trees: An approximation. *Computers & Operations Research*, 23(12):1171–1182, 1996.

[3] C. F. Bazlamaçc and K. S. Hindi. Minimum-weight spanning tree algorithms a survey and empirical study. *Computers & Operations Research*, 28(8):767–785, 2001.

[4] O. Borůvka. O jistém problem minimálním. *Práce moravské přírodovědecké společnosti*, III(3):37–58, 1926.

[5] D. Brookfield, H. Boussabaine, and C. Su. Identifying reference companies using the book-to-market ratio: a minimum spanning tree approach. *The European Journal of Finance*, 19(6):466–490, 2013.

[6] P. Camerini, G. Galbiati, and F. Maffioli. On the complexity of finding multi-constrained spanning trees. *Discrete Applied Mathematics*, 5(1):39–50, 1983.

[7] P. J. S. Cardoso. *Ant colony algorithms for multiple objective combinatorial optimization: applications to the minimum spanning trees problems*. PhD thesis, University of Seville, May 2006.

[8] A. Cayley. A theorem on trees. *Quarterly Journal of Mathematics*, 23:376378, 1889.

[9] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

[10] J. C. N. Clímaco and M. M. B. Pascoal. Multicriteria path and tree problems: discussion on exact algorithms and applications. *International Transactions in Operational Research*, 19(1-2):63–98, 2012.

[11] H. W. Corley. Efficient spanning trees. *Journal of Optimization Theory and Applications*, 45(3):481–485, March 1985.

[12] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23(4):864–894, Aug. 1994.

[13] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani. *Algorithms*, chapter 5, pages 139–167. McGraw-Hill Education, July 2006.

[14] J. Devillers and J. Dore. Heuristic potency of the minimum spanning tree (mst) method in toxicology. *Ecotoxicology and Environmental Safety*, 17(2):227–235, 1989.

[15] L. Di Puglia Pugliese, F. Guerriero, and J. L. Santos. Dynamic programming for spanning tree problems: application to the multi-objective case. *Optimization Letters*, 9(3):437–450, 2015.

[16] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[17] N. Draper and H. Smith. *Applied regression analysis*. John Wiley & Son, Inc., 3 edition, April 1998.

[18] B. Duran and P. Odell. *Cluster Analysis: A Survey*. Lecture Notes in Economics and Mathematical Systems. Springer Berlin Heidelberg, 2013.

[19] J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1(1):127–136, 1971.

[20] M. Ehrgott. *Multicriteria optimization*. Springer-Verlag, Berlin Heidelberg, 2005. Originally published as volume 491 in the series: Lecture Notes in Economics and Mathematical Systems.

[21] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, 22(4):425–460, 2000.

[22] M. Ehrgott and X. Gandibleux. Multiobjective combinatorial optimization — theory, methodology, and applications. In M. Ehrgott and X. Gandibleux, editors, *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, pages 369–444. Springer US, Boston, MA, 2002.

[23] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the Association for Computing Machinery*, 34(3):596–615, July 1987.

[24] H. N. Gabow. Two algorithms for generating weighted spanning trees in order. *SIAM Journal on Computing*, 6(1):139–150, 1977.

[25] L. Georgiadis. Arborescence optimization problems solvable by edmonds algorithm. *Theoretical Computer Science*, 301(1):427–437, 2003.

[26] D. Gombos, J. A. Hansen, J. Du, and J. McQueen. Theory and applications of the minimum spanning tree rank histogram. *Monthly Weather Review*, 135:1490–1505, April 2007.

[27] L. Gouveia, L. Simonetti, and E. Uchoa. Modeling hop-constrained and diameter-constrained minimum spanning tree problems as steiner tree problems over layered graphs. *Mathematical Programming*, 128(1):123–148, 2011.

[28] R. L. Graham and P. Hell. On the history of the minimum spanning tree problem. *IEEE Annals of the History of Computing*, 7(1):43–57, January 1985.

[29] F. Guerriero and R. Musmanno. Label correcting methods to solve multicriteria shortest path problems. *Journal of Optimization Theory and Applications*, 111(3):589–613, 2001.

[30] A. Gupta. Lecture 1: Deterministic MSTs. `http://www.cs.cmu.edu/~anupamg/advanced/lectures/lecture01.pdf`, January 2015. Accessed: 2015-12-09.

[31] H. W. Hamacher and G. Ruhe. On spanning tree problems with multiple objectives. *Annals of Operations Research*, 52(4):209–230, 1994.

[32] L. Han and Y. Wang. A novel genetic algorithm for multi-criteria minimum spanning tree problem. In Y. Hao, J. Liu, Y. Wang, Y.-m. Cheung, H. Yin, L. Jiao, J. Ma, and Y.-C. Jiao, editors, *Computational Intelligence and Security: International Conference, CIS 2005, Xi'an, China, December 15-19, 2005, Proceedings Part I*, pages 297–302. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[33] V. Jarník. O jistém problem minimálním. *Práce moravské přírodovědecké společnosti*, VI(4):57–63, 1930.

[34] D. R. Karger, P. N. Klein, and R. E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *Journal of the Association for Computing Machinery*, 42(2):321–328, March 1995.

[35] J. D. Knowles and D. W. Corne. A comparison of encodings and algorithms for multiobjective minimum spanning tree problems. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, volume 1, pages 544–551, 2001.

[36] S. O. Krumke and H.-C. Wirth. On the minimum label spanning tree problem. *Information Processing Letters*, 66(2):81–85, 1998.

[37] J. B. Kruskal. On the shortest spanning subtree of a graph and the travelling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, February 1956.

[38] Y. Li, C. Y. Zou, S. Zhang, and M. I. Vai. Research on multi-objective minimum spanning tree algorithm based on ant algorithm. *Research Journal of Applied Sciences, Engineering and Technology*, 5(21):5051–5056, 2013.

[39] R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[40] F. Meyer and L. Najman. *Segmentation, Minimum Spanning Tree and Hierarchies*, chapter 9, pages 229–261. John Wiley & Sons, Inc., 2013.

[41] M. D. Moradkhan and W. N. Browne. A knowledge-based evolution strategy for the multi-objective minimum spanning tree problem. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1391–1398, 2006.

[42] J. Nešetřil and H. Nešetřilová. The origins of minimal spanning tree algorithms - Borůvka and Jarník. *Documenta Mathematica*, Extra Volume ISMP:127–141, 2012.

[43] T. Öncan. Design of capacitated minimum spanning tree with uncertain cost and demand parameters. *Information Sciences*, 177(20):4354–4367, 2007.

[44] J. M. Paixão and J. L. Santos. Labeling methods for the general case of the multi-objective shortest path problem – a computational study. In A. Madureira, C. Reis, and V. Marques, editors, *Computational Intelligence and Decision Making: Trends and Applications*, pages 489–502. Springer Netherlands, Dordrecht, 2013.

[45] S. Pettie and V. Ramachandran. An optimal minimum spanning tree algorithm. *Journal of the Association for Computing Machinery*, 49(1):16–34, January 2002.

[46] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, November 1957.

[47] R. Ramos, S. Alonso, J. Sicilia, and C. Gonzlez. The problem of the optimal biobjective spanning tree. *European Journal of Operational Research*, 111(3):617–628, 1998.

[48] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi. Spanning treesshort or small. *SIAM Journal on Discrete Mathematics*, 9(2):178–200, 1996.

[49] S. Ruzika and H. W. Hamacher. A survey on multiple objective minimum spanning tree problems. In J. Lerner, D. Wagner, and K. A. Zweig, editors, *Algorithmics of Large and Complex Networks*, pages 104–116. Springer-Verlag, Berlin, Heidelberg, 2009.

[50] H. F. Salama, D. S. Reeves, and Y. Viniotis. The delay-constrained minimum spanning tree problem. In *Proceedings Second IEEE Symposium on Computer and Communications*, pages 699–703, Jul 1997.

[51] A. K. S. Sanger and A. K. Agrawal. Comparison of tree encoding schemes for biobjective minimum spanning tree problem. In *2010 2nd IEEE International Conference on Information and Financial Engineering*, pages 233–236, Sept 2010.

[52] P. M. Spira and A. Pan. On finding and updating spanning trees and shortest paths. *SIAM Journal on Computing*, 4(3):375–380, 1975.

[53] S. Steiner and T. Radzik. Computing all efficient solutions of the biobjective minimum spanning tree problem. *Computers & Operations Research*, 35(1):198–211, 2008. Part Special Issue: Applications of {OR} in Finance.

[54] K. J. Supowit, D. A. Plaisted, and E. M. Reingold. Heuristics for weighted perfect matching. In *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, STOC '80, pages 398–419, New York, NY, USA, 1980. ACM.

[55] M. Thorup. Near-optimal fully-dynamic graph connectivity. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, pages 343–350, New York, NY, USA, 2000. ACM.

[56] J. A. Torkestani. Degree constrained minimum spanning tree problem: a learning automata approach. *The Journal of Supercomputing*, 64(1):226–249, 2013.
[57] C.-C. Wei, S.-Y. Hsieh, C.-W. Lee, and S.-L. Peng. An improved approximation algorithm for the partial-terminal steiner tree problem with edge cost 1 or 2. *J. of Discrete Algorithms*, 35(C):62–71, Nov. 2015.
[58] Wikipedia. Minimum spanning tree.
`https://en.wikipedia.org/wiki/Minimum_spanning_tree`.
Accessed: 2015-12-09.
[59] B. Y. Wu and K.-M. Chao. *Spanning Trees and Optimization Problems*. Discrete Mathematics and Its Applications. Chapman and Hall/CRC, Abingdon, January 2004.
[60] G. Zhou and M. Gen. Genetic algorithm approach on multi-criteria minimum spanning tree problem. *European Journal of Operational Research*, 114(1):141–152, 1999.
[61] L. Zsak. Variations for spanning trees. *Annales Mathematicae et Informaticae*, 33:151165, 2006.

José Luis Santos
CMUC, Department of Mathematics, University of Coimbra, Apartado 3008, 3001–501 Coimbra, Portugal
E-mail address: `zeluis@mat.uc.pt`

Luigi Di Puglia Pugliese
Department of Mechanical, Energy and Management Engineering, University of Calabria, Italy
E-mail address: `luigi.dipugliapugliese@unical.it`

Francesca Guerriero
Department of Mechanical, Energy and Management Engineering, University of Calabria, Italy
E-mail address: `francesca.guerriero@unical.it`