Minimum Complete Pareto Front of a Biobjective Minimum Spanning Trees Problem

Lasko Laskov
Department of Informatics
New Bulgarian University
Sofia, Bulgaria
llaskov@nbu.bg

Marin Marinov
Department of Informatics
New Bulgarian University
Sofia, Bulgaria
mlmarinov@nbu.bg

***Abstract*—We propose an exact method that finds a minimum complete Pareto front of the biobjective minimum length minimum risk spanning trees problem. The method consists in two algorithms. The first algorithm finds a single minimum length spanning tree that has minimum risk in running time complexity $O(m + n\lg n)$. The second algorithm calculates a minimum complete Pareto front with an algorithm that has a pseudo-polynomial complexity.**

We present the mathematical theorems that proof the correctness of both algorithms, and we show their computational complexity. Also, we describe numerical experiments that illustrate their implementation.

***Index Terms*—optimization in graphs, minimum spanning trees, Pareto optimality, biobjective optimization**

I. Introduction

In its standard, single-objective form the *minimum spanning tree* (MST) problem is the problem of finding a single acyclic subset of the edges of a connected, undirected, weighed graph with a minimized total weight [1]. It is a classical problem in field of graph algorithms and combinatorial optimization that has numerous practical applications. Besides its obvious usage in the design of computer, communication, telecommunication and transportation networks [2], it is used in the solution of other combinatorial optimization problems such as the traveling salesman problem [3], and it plays important role in image processing, speech recognition, pattern recognition and clustering algorithms [4].

The single-objective MST problem can be solved in polynomial running time complexity and three well-known greedy algorithms exist: Borůvka's algorithm (see for example [3]), Kruskal's algorithm [5] and Prim's algorithm [6]. The earliest of them, the Borůvka's algorithm has been published in 1926, but more recently gained attention due to the possibility of parallel implementation [7]. We also have to mention that the Prim's algorithm share the same greedy principle as the Dijkstra's algorithm [8], and that in the latter work, besides the single-source shortest path problem, Dijkstra proposes an algorithm that solves the MST problem as well.

Besides the central role of the single-objective MST problem in combinatorial optimization, many problems from practice require the solution of its multi-objective and particularly, its biobjective version. A classical example of such application is the design of computer networks (see for example [9]). However, the addition of an extra restriction or objective function to the problem causes its exponential complexity and it is reported to be NP-hard (see [10], [11], [12], [13]). Something more, in the presence of more than one objective function, it is not possible to find a single solution that optimizes all of them, and we are looking for a set of non-dominating solutions, called *Pareto optimal set* or *Pareto front*. In this case we are not looking for a single MST but for a whole set of MSTs.

Because of the complexity of the biobjective MSTs problem, often in the specialized literature it is approached using heuristic and approximation methods. In [11] authors present a memetic algorithm that represents a genetic algorithm hybridized with a tabu search procedure. In [12] a heuristic is proposed that speeds up a k-best method and it is compared with two branch-and-bound schemes. In [13] a metaheuristic approach is proposed that solves biobjective spanning trees with minimum total cost and minimum diameter problem. The described solution is based on multi-objective evolutionary algorithm and on a nondominated sorting genetic algorithm.

Since some applications may require exact solutions, there are also a number of works that solve the biobjective MSTs problem using exact methods. In [10] a two phase method is proposed, based on an extension of the Kruskal's algorithm and a branch-and-bound procedure. In [14] the authors solve the biobjective spanning trees with minimum total cost and minimum diameter problem. The proposed approach finds the Pareto front of the problem and is based on exact method, with both correctness and running time verified experimentally. In [15] is described a two-phase method that finds the complete set of Pareto solutions. In the first phase three different strategies are used, including Prim's greedy algorithm incorporated in a weighted sum approach. The second phase generates all the spanning trees using a recursive method.

In this paper we propose an exact method that finds the minimum complete Pareto front of a version of biobjective MSTs problem with first objective function being minimum length, and the second objective function being minimum risk. Our method consists of two algorithms. The first algorithm, described in Sec. III, is an extension of the Prim's greedy strategy and finds a single Pareto optimal spanning tree with minimum length and minimum risk in running time complexity $O(m + n \lg n)$. It is used as an intermediate stage of each iteration of the main algorithm, given in Sec. IV, that constructs the minimum Pareto front in pseudo-polynomial computational complexity.

II. Problem formulation

We denote the connected undirected graph $G = (V, E)$ with $n = |V|$ number of vertices and $m = |E|$ number of edges. We assume that the set of vertices is $V = \{1, 2, \ldots, n\}$ and the set of edges is $E \subseteq V^2$.

Two objective functions f and g are given. The function $f : E \to \mathbb{R}_+$ assigns to each edge $(u, v) \in E$ the positive real number $f(u, v)$, which denotes the *length* of the edge (u, v). The function $g : E \to \mathbb{R}_+$ assigns to each edge $(u, v) \in E$ the positive real number $g(u, v)$, which denotes the *risk* of the edge (u, v).

The graph G together with the two objective functions f and g form the network $N = (V, E, f, g)$. In the following algorithms the network is given by its adjacency lists representation:

$$N.Adj = [\langle (v, f(u, v), g(u, v)), \ldots \rangle, \ldots]. \quad (1)$$

For each subset of edges $A \subseteq E$ we define the numbers $x(A)$ (see Eq. (2)) and $y(A)$ (see Eq. (3)).

$$x(A) = \sum_{i=1}^{k} f(u_i, v_i) \quad (2)$$

$$y(A) = \max\{g(u_i, v_i) : i \in \{1, 2, \ldots, k\}\} \quad (3)$$

The number $x(A)$ denotes the *length*, while the number $y(A)$ denotes the *risk* of the subset of edges A.

With $\mathcal{T}$ we denote the set of all spanning trees of the network N.

Definition 1: We call the tree $t' \in \mathcal{T}$ *Pareto optimal* when there does not exist another tree $t \in \mathcal{T}$, for which any of the following two conditions is fulfilled:

- $x(t) < x(t')$ and $y(t) \leq y(t')$;
- $x(t) \leq x(t')$ and $y(t) < y(t')$.

Definition 2: We will say that the tree t' and t are *equivalent* and we will denote it with $t' \sim t$, when $x(t') = x(t)$ and $y(t') = y(t)$.

Definition 3: We will say that the tree t is *dominated* by the tree t' and we will denote it $t \prec t'$, when $x(t') < x(t)$ and $y(t') \leq y(t)$ or $x(t') \leq x(t)$ and $y(t') < y(t)$.

Let the set of all Pareto optimal trees in the network N is denoted by P. Then:

$$P = \bigcup_{i=1}^{K} P_i, \quad (4)$$

where P_i denotes the classes of equivalent Pareto optimal MSTs, and K is the number of such classes.

Definition 4: We call a *minimum complete Pareto front* each set $M = \{t_1, t_2, \ldots, t_K\}$, for which $t_i \in P_i$, $\forall i \in \{1, 2, \ldots, K\}$.

We will use the following relations in the set of edges E. Let e_1 and e_2 are two arbitrary edges in the network N.

Definition 5: We will say that the edge e_1 is *equivalent* to the edge e_2, and we will denote it $e_1 \sim e_2$, when $f(e_1) = f(e_2)$ and $g(e_1) = g(e_2)$.

Definition 6: We will say that the edge e_2 is *dominated* by the edge e_1, and we will denote it $e_2 \prec e_1$, when one of the following two conditions is fulfilled:

- $f(e_1) < f(e_2)$, or
- $f(e_1) = f(e_2)$ and $g(e_1) < g(e_2)$.

For any two arbitrary selected edges e_1 and e_2, exactly one of the following three relations hold: (i) $e_2 \sim e_1$; (ii) $e_2 \prec e_1$; (iii) $e_1 \prec e_2$. Besides that, when one of the two $e_2 \sim e_1$ or $e_2 \prec e_1$ is fulfilled, we will write $e_2 \precsim e_1$.

It can be easily verified that any of the relations $e_2 \prec e_1$ or $e_2 \precsim e_1$ establishes a *linear order* on the set E.

Let $B \subset E$. We call the edge e_0 from B a *dominant* of B, when for each edge e from B is fulfilled $e \precsim e_0$.

We will denote the set of all dominants of the set B by $Dom\{B\}$.

The following two properties hold.

Property 1: Each nonempty subset of E has at least one dominant.

Property 2: If $e_1 \in Dom\{B\}$ and $e_2 \in Dom\{B\}$, then $e_1 \sim e_2$.

The goal of this paper is to solve the following problem.

Problem 1 (Main problem): Given the network N, find a minimum complete Pareto front of the biobjective MSTs problem with first objective function being minimum length, and the second objective function being minimum risk.

In the solution of the main problem we will use the solution of the following helper problem.

Problem 2 (Helper problem): Given the network N, find a single Pareto optimal spanning tree A with minimum length and minimum risk.

III. Pareto optimal spanning tree with minimum length and minimum risk

In this section we will present the solution of the helper problem (given in Problem 2). The algorithm that we describe, in analogy to the Prim's algorithm [6], uses an inductive procedure to consecutively discover the edges of a Pareto optimal spanning tree A with minimum length and minimum risk. The procedure is implemented using a *greedy-choice property*, which in our case is based on the linear order of the edges of the network N, which is given in Sec. II.

In Algorithm 1 the tree A is represented by an n-component vector p that stores the list of parents of the vertices of the tree. When $A = \varnothing$, then p is a zero vector, and when an edge (u, v) is assigned to A, the vertex u is parent of the vertex v and $p[v] = u$. We will define the set of edges that we will assign to A by traversing the set of vertices V. We denote by Q the priority queue of vertices that are not yet traversed. Initially, $Q = V$. We denote by U the set of traversed vertices. Apparently, $U = V \setminus Q$.

We define an arbitrary vertex r as the root of the tree. After that, $U = \{r\}$ and $Q = V \setminus U$. A minimum length spanning tree with minimum risk can be constructed using the following procedure.

Procedure 1: Inductive procedure to construct a minimum length spanning tree with minimum risk.

I. Base step.

1) Since the number of elements of Q is finite, there exists a vertex v_1 in Q, such that

$$(r, v) \precsim (r, v_1), \text{ for each vertex } v \in Q. \quad (5)$$

2) We update the sets Q and U by removing the vertex v_1 from Q, and assigning it to U.
3) We assign the edge (r, v_1) to A by storing $p[v_1] = r$.

II. Inductive step.

1) Since the number of elements of the sets Q and U are finite, there exist vertices $v_1 \in Q$ and $u_1 \in U$, such that

$$(u, v) \precsim (u_1, v_1), \text{ for each } u \in U \text{ and } v \in Q. \quad (6)$$

2) We move the vertex v_1 from Q to the set U and get the new U and Q.
3) We assign the edge (u_1, v_1) to A by storing $p[v_1] = u_1$.
4) If $Q \neq \varnothing$, go to step II.1. Otherwise, stop.

After the execution of Procedure 1, the set A is a minimum length spanning tree with minimum risk. Analogous to the proof of the correctness of Prim's algorithm, the correctness of the procedure is proved here.

Theorem 1: The set A constructed by Procedure 1 is a minimum length spanning tree with minimum risk.

The proof of Theorem 1 follows from the fact that Procedure 1 expands the set A in such way that it is a subset of a minimum length spanning tree T with minimum risk.

The implementation of Procedure 1 by Algorithm 1 uses the helper Boolean function $\textsc{Best}(a, b)$. It assigns to the pairs of numbers $a = (a_1, a_2)$ and $b = (b_1, b_2)$ the value $true$ if one of the following two conditions is met: (i) $a_1 < b_1$ or (ii) $a_1 = b_1$ and $a_2 < b_2$. When both conditions are not satisfied, the value that is returned by the function is $false$.

The algorithm maintains the n-component vector d that stores key attributes for each vertex. In the standard single-objective Prim's algorithm the role of a key $d[v]$ for a given vertex v is to denote the minimum weight that is associated to an edge that connects v to any vertex in the tree A. In the examined case we have two objective functions, and each component of the vector d is composed by the pair $(f(u, v), g(u, v))$, where $(u, v) \in Dom\{E_1\}$, $E_1 = \{(u, v) \in E : u \in U\}$. If there is no such edge, then $E_1 = \varnothing$ and $d[v] = (\infty, \infty)$. In this way, the pair stored in $d[v]$ characterizes the distance of the unvisited vertex v from the set U of traversed vertices.

The min-priority queue Q is keyed by the components of the vector d. The algorithm maintains the min-priority queue using the functions $\textsc{Insert}(Q, v)$, $\textsc{Extract-Min}(Q)$ and $\textsc{Decrease-Key}(Q, w, (x, y))$. The functions $\textsc{Insert}(Q, u)$ insets in the priority queue a vertex u. The purpose of the

Algorithm 1 Function $\textsc{LenthRisk}(N.Adj, r)$

```
Input: adjacency lists $N.Adj$ and root vertex $r$
Output: minimum spanning tree with minimum risk $p$
1: $d[r] \leftarrow (0, 0)$
2: $Q \leftarrow \varnothing$
3: for each vertex $u \in N.V$ do
4:     $\textsc{Insert}(Q, u)$
5: end for
6: while $Q \neq \varnothing$ do
7:     $v \leftarrow \textsc{Extract-Min}(Q)$
8:     $b[v] \leftarrow true$
9:     for $i \leftarrow 1$ to $|N.Adj[v]|$ do
10:        $(w, x, y) \leftarrow N.Adj[v][i]$
11:        if $b[w] = false$ and $\textsc{Best}((x, y), d[w])$ then
12:            $d[w] \leftarrow (x, y)$
13:            $\textsc{Decrease-Key}(Q, w, (x, y))$
14:            $p[w] \leftarrow v$
15:        end if
16:    end for
17: end while
18: return $p$
```

$\textsc{Extract-Min}(Q)$ is to return a dominant vertex that is currently stored in Q, and to remove it from the queue. The function $\textsc{Decrease-Key}(Q, w, (x, y))$ updates the corresponding attribute $d[w]$ of the vertex w with the new pair of values (x, y), and also updates the intrinsic structure of the priority queue. To ensure an efficient implementation of the abstract data type min-priority queue, we adopt Fibonacci heap [16], which guarantees that the computational complexity of $\textsc{Extract-Min}(Q)$ is $O(\lg n)$ and that $\textsc{Decrease-Key}(Q, w, (x, y))$ has constant computational complexity.

In Algorithm 1 we will represent whether a vertex v belongs to the set U of traversed vertices by the n-component Boolean vector b, given in Eq. (7).

$$b[v] = \begin{cases} true, & \text{if } v \in U \\ false, & \text{if } v \in Q \end{cases} \quad (7)$$

On the initial state of Algorithm 1 the vector p is the n-component zero vector and all components of the vector b are set to $false$. The input of the algorithm are the adjacency lists $N.Adj$ of the network and the arbitrary selected root r of the tree that will be constructed.

The following two theorems hold.

Theorem 2: After the termination of Algorithm 1 in the vector p is stored a minimum length spanning tree with minimum risk.

The proof of Theorem 2 follows directly from Theorem 1.

Theorem 3: The computational complexity of Algorithm 1 is $O(m + n \lg n)$.

The proof of Theorem 3 follows directly from the complexity of $\textsc{Extract-Min}(Q)$ and $\textsc{Decrease-Key}(Q, w, (x, y))$ functions mentioned above, and from the complexity of the Prim's algorithm with min-priority queue implemented using Fibonacci heap (see [17]).

Algorithm 2 Function MCFP$(N.Adj, r)$

```
Input: adjacency lists $N.Adj$ and root vertex $r$
Output: a minimum complete Pareto front $F$ and its corresponding weights $W$
1: $F \leftarrow \varnothing$, $W \leftarrow \varnothing$
2: while CONNECTED$(N.Adj) = true$ do
3:     $p \leftarrow$ LENTHRISK$(N.Adj, r)$
4:     $(T, length, risk) \leftarrow$ OPTTREE$(p)$
5:     $F \leftarrow F \cup \{T\}$, $W \leftarrow W \cup \{(length, risk)\}$
6:     $N.Adj \leftarrow$ RESTRICTED$(N.Adj, risk)$
7: end while
8: return $\{F, W\}$
```

Corollary 1: Let T_0 be the minimum length spanning tree with minimum risk constructed by Algorithm 1. Then T_0 is a Pareto optimal tree for the biobjective minimum length minimum risk spanning trees problem.

Proof: Let p is the vector that stores the list of parents of the tree T_0 that is returned by Algorithm 1. From Theorem 2 it follows that for each spanning tree T the statements holds:

1) $x(T_0) \leq x(T)$;
2) if $x(T_0) = x(T)$ then $y(T_0) \leq y(T)$.

We assume that there exists a spanning tree T_1 that dominates T_0. From statement 1) follows that $x(T_0) = x(T_1)$. Then, from the assumption it follows that $y(T_1) < y(T_0)$, which contradicts statement 2).

The resulting contradiction proves that there does not exist a spanning tree that dominates T_0. ■

IV. The minimum complete Pareto front problem

In this section we will solve the main problem given in Problem 1 using Algorithm 2. Our solution uses the function LENTHRISK$(N.Adj, r)$ defined in Algorithm 1 in the previous section. The presented algorithm uses three more functions: OPTTREE(p), CONNECTED$(N.Adj)$ and RESTRICTED$(N.Adj, risk)$.

Let p be the resulting list of parents that represents the tree constructed by LENTHRISK$(N.Adj, r)$. We define the helper function OPTTREE(p) that returns the triple $(T, length, risk)$, where T is the set of edges of the tree stored in p, $lenght = x(T)$ and $risk = y(T)$. The computational complexity of this function is $O(m)$.

The helper Boolean function CONNECTED$(N.Adj)$ returns $true$ if the network N is connected, and returns $false$ otherwise. It is implemented using breadth first search of N and therefore it has complexity $O(n + m)$ (see [1]).

The helper function RESTRICTED$(N.Adj, risk)$ takes as an input the adjacency lists $N.Adj$ and the boundary $risk$. It traverses the edges of N and defines the subnetwork $N_1 = (V, E_1, f, g)$, where $E_1 = \{(u, v) \in E : g(u, v) < risk\}$. The computational complexity of this function is $O(m)$.

Algorithm 2 takes as an input the adjacency lists $N.Adj$ and a selected vertex r that will be the root of the trees that are going to be constructed. The result of the calculations is stored in lists F and W. The list F stores a minimum complete Pareto optimal front as a sequence of Pareto optimal MSTs. For each Pareto optimal MST $T \in F$ the list W stores its corresponding weights as a $(x(T), y(T))$ pair.

Theorem 4: After the termination of Algorithm 2, the list F contains a minimum complete Pareto front of the biobjective minimum length minimum risk spanning trees problem.

The proof of Theorem 4 follows from Corollary 1 and the correctness of the helper functions used in Algorithm 2.

Theorem 5: The computational complexity of Algorithm 2 is $O(K(m + n \lg n))$, where K is the number of classes of equivalent Pareto optimal MSTs.

Theorem 5 follows from the complexity of Algorithm 1 and the fact that the **while** loop of the Algorithm 2 performs K number of iterations.

The following Example 1 illustrates the execution of Algorithm 2.

Example 1: We will solve the main problem formulated in Problem 1 for the network N_1 given with the following adjacency lists:

$$\begin{aligned} N_1.Adj = [&\langle(2,7,10),(3,7,6),(4,9,4),(5,15,8)\rangle,\\ &\langle(1,7,10),(3,15,4),(4,7,8),(5,9,6)\rangle,\\ &\langle(1,7,6),(2,15,4),(4,7,8),(5,9,4)\rangle,\\ &\langle(1,9,4),(2,7,8),(3,7,8),(5,9,6)\rangle,\\ &\langle(1,15,8),(2,9,6),(3,9,4),(4,9,6)\rangle]. \end{aligned} \tag{8}$$

Solution. For concreteness, let us select the vertex $r = 1$ as the root of the trees that are going to be constructed.

In the first row of the algorithm the two empty lists F and W are constructed. Then the algorithm enters the **while** loop and evaluates its condition. Since the network N_1 is connected, CONNECTED$(N_1.Adj)$ returns $true$ and the loop starts its *first iteration.*

The function LENTHRISK$(N.Adj, r)$ calculates the list of parents of a Pareto optimal tree with minimum length and minimum risk $p = [0, 4, 1, 3, 3]$. The correctness of this step follows from Theorem 2.

The function OPTTREE(p) from p calculates the triple $(T_0, length, risk)$, where $T_0 = \{(4,2),(1,3),(3,4),(3,5)\}$, $length = 30$ and $risk = 8$. The correctness of this result follows from the correctness of OPTTREE(p), while in this particular case it can be verified directly.

In line 5 the tree T_0 is included in the list F and its corresponding weights are included in the list W and their current state is $F = \langle\{(4,2),(1,3),(3,4),(3,5)\}\rangle$ and $W = \langle(30,8)\rangle$. Corollary 1 verifies that the included element T_0 in F is a Pareto optimal tree of the initial network N_1.

After that, the function RESTRICTED$(N.Adj, 8)$ modifies the adjacency lists of the network which results in (9).

$$\begin{aligned} N_1'.Adj = [&\langle(3,7,6),(4,9,4)\rangle,\langle(3,15,4),(5,9,6)\rangle,\\ &\langle(1,7,6),(2,15,4),(5,9,4)\rangle,\langle(1,9,4),(5,9,6)\rangle,\\ &\langle(2,9,6),(3,9,4),(4,9,6)\rangle]. \end{aligned} \tag{9}$$

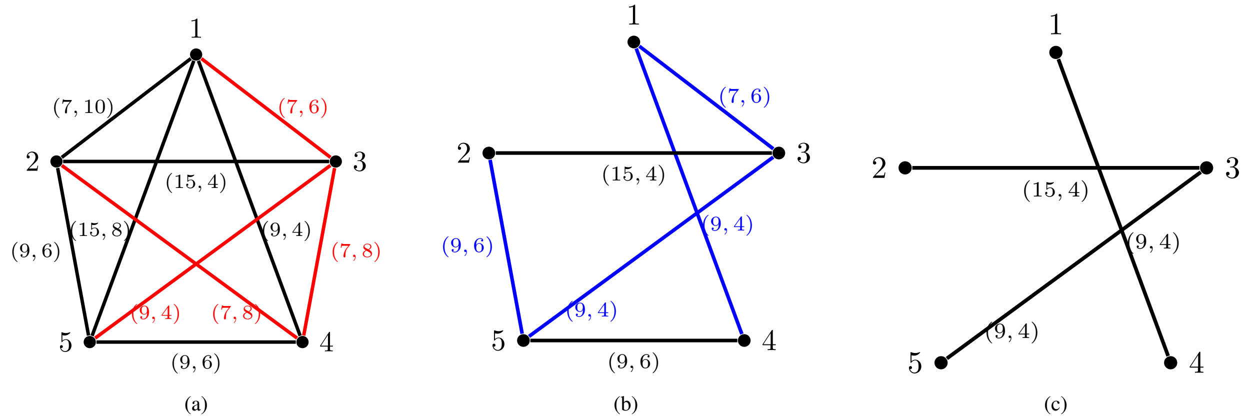

Fig. 1. (a) The example network N_1 given in Eq. (8) with the Pareto optimal MST T_0 marked in red; (b) the restricted network N_1' given in Eq. (9) with the Pareto optimal MST T_1 marked in blue; (c) the restricted network N_1'' given in Eq. (11) is not connected and stops the algorithm

The adjacency lists (9) represents the subnetwork N_1' that has the same set of vertices $V\{1,2,3,4,5\}$ as the network N_1. The only difference between N_1' and N_1 is that N_1' contains only these edges from N_1 that have risk strictly less than 8. Therefore, if T is a spanning tree in N_1 with risk less than 8, then it is a spanning tree in the subnetwork N_1'. And vice versa, if T is a spanning tree in the subnetwork N_1' if is also a spanning tree in the network N_1.

In the condition of the **while** loop it is verified that the network N_1' is connected and the loop proceeds to its *second iteration*.

The calculations of the second iteration of the **while** loop are completely analogous to the calculations of the first iteration. The function LENTHRISK$(N.Adj, r)$ calculates the list of parents $p' = [0, 5, 1, 1, 3]$ of a Pareto optimal tree with minimum length and minimum risk in N_1'. The function OPTTREE(p') stores the discovered tree as a list of edges $T_1 = \{(5,2),(1,3),(1,4),(3,5)\}$, and calculates its length $x(T_1) = 34$ and risk $y(T_1) = 6$. The correctness of these results is established by repeating the proof from the first iteration of the loop.

The lists F and W are updated and are given in (10).

$$F = \langle T_0, T_1\rangle, W = \langle (30,8),(34,6)\rangle, \tag{10}$$

where

$$T_0 = \{(4,2),(1,3),(3,4),(3,5)\},$$

$$T_1 = \{(5,2),(1,3),(1,4),(3,5)\}.$$

The function RESTRICTED$(N'.Adj, 6)$ modifies the adjacency lists of N' and the result is given in Eq. (11).

$$\begin{gathered} N_1''.Adj = [\langle (4,9,4)\rangle, \langle (3,15,4)\rangle, \\ \langle (2,15,4),(5,9,4)\rangle, \langle (1,9,4)\rangle, \langle (3,9,4)\rangle] \end{gathered} \tag{11}$$

The condition of the **while** loop evaluates to $false$ because the network N_1'' is not connected, and the loop stops. Algorithm 2 returns as result the two lists given in Eq. (10). F contains a minimum Pareto front of the biobjective MSTs problem which follows from Theorem 4. In this particular case this can be easily verified directly. ■

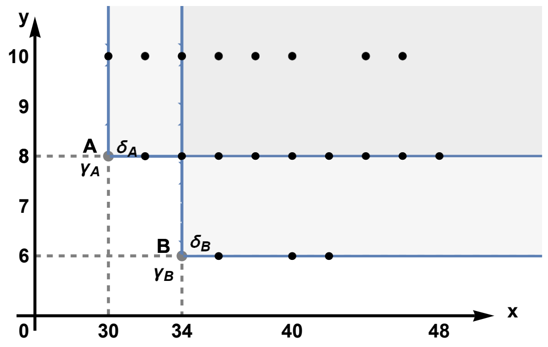

Fig. 2. Minimum complete Pareto front of Example 1 with x denoting the length and y denoting the risk. The points A and B represent the two classes of equivalent Pareto optimal MSTs, while the black points represent the remaining equivalent spanning trees in N. The two pairs of vertically opposite angles γ_A, δ_A and γ_B, δ_B are given, with δ_A and δ_B containing the corresponding dominated spanning trees.

Fig. 2 shows the numerical calculations of Example 1 and it clearly shows that T_0 and T_1 cannot be compared, and all other spanning trees are dominated by at least one of them.

V. EXPERIMENTS AND COMPUTATIONAL EFFICIENCY

We have conducted numerical experiments in order to verify our implementations of Algorithm 1 and Algorithm 2, and also to investigate the effectiveness of the proposed method. We have compared our algorithms with implementations based on brute-force generation of all spanning trees, and for relatively small networks both approaches immediately produce identical results. However, even for a network with $n = 10$ number of vertices, the brute-force approach is not applicable any more, because the number of spanning trees grows exponentially.

We have verified the effectiveness of our implementations with a series of experiments with random networks with number of vertices starting from $n = 10$ up to $n = 10^2$

and number of edges $m = n(n-1)/2$. We will point that Algorithm 2 shows stable behavior even for networks with bigger n and m.

TABLE I
NUMBER OF CLASSES OF EQUIVALENT PARETO OPTIMAL MSTS K DEPENDS ON THE NUMBER OF POSSIBLE LENGTH WEIGHTS m_1 AND THE NUMBER OF POSSIBLE RISK WEIGHTS m_2

Net. size	**Restricted weights**			**Varied weights**		
n	m_1	m_2	K	m_1	m_2	K
40	156	8	8	9900	4950	91
60	354	18	17	9900	4950	152
80	632	32	30	9900	4950	213
100	990	50	45	9900	4950	298

In the conducted experiments the weights of the edges are randomly generated. The length of each edge is randomly selected from m_1 different natural numbers and the risk of each edge is randomly selected from m_2 different natural numbers. As shown in Table I, the number of number of classes of equivalent Pareto optimal MSTs K depends on how restricted or varied are the sets of possible length and risk weights.

In the restricted weights case the number of edges with different length weights does not exceed 20% of all edges, and the number of edges with different risk weights does not exceed 1% of all edges. On contrary, in the case of varied weights, the weights of the edges are selected from more than $\max\{n, m\}$ different natural numbers.

The exponential grow of the examined problem is expressed with K, and for huge networks with varied weights we can easily modify the algorithm to restrict the number of discovered number of classes of equivalent Pareto optimal MSTs F_0 to a predefined constant K_0, reducing the running time of the algorithm to polynomial complexity. In this case for the constructed list F_0 we have that $F_0 \subseteq F$. Besides that, we acquire the information whether the minimal complete Pareto front has more than $K_0 - 1$ elements and whether $F_0 = F$.

VI. CONCLUSION

In this paper we propose an exact method that constructs a minimum complete Pareto front of the biobjective MSTs problem with first objective function being the minimum length and the second objective function being the minimum risk. Algorithm 2 that finds a minimum complete Pareto front has complexity $O(K(m + n \lg n))$, where K is the number of classes of equivalent Pareto optimal MSTs.

The solution of the main problem is based on the solution of the helper problem to construct a single Pareto optimal spanning tree with minimal length and minimal risk. This problem itself is a subject of research in the field. The algorithm that solves it is an extension of the Prim's algorithm [6] that uses greedy-choice property based on the linear order of the edges of the network that we describe in Sec. II. The running time complexity of Algorithm 1 that solves the helper problem is $O(m + n \lg n)$.

REFERENCES

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 4th ed. MIT Press, 2022.

[2] R. L. Graham and P. Hell, "On the history of the minimum spanning tree problem," *Annals of the History of Computing*, vol. 7, no. 1, pp. 43–57, 1985.

[3] C. F. Bazlamaçcı and K. S. Hindi, "Minimum-weight spanning tree algorithms a survey and empirical study," *Computers & Operations Research*, vol. 28, no. 8, pp. 767–785, 2001.

[4] M. Gagolewski, A. Cena, M. Bartoszuk, and Łukasz Brzozowski, "Clustering with minimum spanning trees: How good can it be?" *Journal of Classification*, vol. 41, 2024.

[5] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical society*, vol. 7, no. 1, pp. 48–50, 1956.

[6] R. C. Prim, "Shortest connection networks and some generalizations," *The Bell System Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957.

[7] D. A. Bader and G. Cong, "Fast shared-memory algorithms for computing the minimum spanning forest of sparse graphs," *Journal of Parallel and Distributed Computing*, vol. 66, no. 11, pp. 1366–1378, 2006.

[8] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[9] L. Cheng, J. Niu, C. Luo, L. Shu, L. Kong, Z. Zhao, and Y. Gu, "Towards minimum-delay and energy-efficient flooding in low-duty-cycle wireless sensor networks," *Computer Networks*, vol. 134, no. 1, pp. 66–77, 2018.

[10] R. M. Ramos, S. Alonso, J. Sicilia, and C. González, "The problem of the optimal biobjective spanning tree," *European Journal of Operational Research*, vol. 111, no. 3, pp. 617–628, 1998.

[11] D. A. M. Rocha, E. F. G. Goldbarg, and M. C. Goldbarg, "A memetic algorithm for the biobjective minimum spanning tree problem," in *Evolutionary Computation in Combinatorial Optimization*, J. Gottlieb and G. R. Raidl, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 222–233.

[12] S. Steiner and T. Radzik, "Computing all efficient solutions of the biobjective minimum spanning tree problem," *Computers & Operations Research*, vol. 35, no. 1, pp. 198–211, 2008.

[13] A. C. Santos, D. R. Lima, and D. J. Aloise, "Modeling and solving the bi-objective minimum diameter-cost spanning tree problem," *Journal of Global Optimization*, vol. 60, pp. 195–216, 2014.

[14] E. G. de Sousa, A. C. Santos, and D. J. Aloise, "An exact method for solving the bi-objective minimum diameter-cost spanning tree problem," *RAIRO-Oper. Res.*, vol. 49, no. 1, pp. 143–160, 2014.

[15] L. Amorosi and J. Puerto, "Two-phase strategies for the bi-objective minimum spanning tree problem," *International Transactions in Operational Research*, vol. 29, no. 6, pp. 3435–3463, 2022.

[16] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *Journal of the ACM*, vol. 34, no. 3, pp. 596–615, July 1987.

[17] B. Korte and J. Vygen, *Spanning Trees and Arborescences*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, pp. 133–157.