Dynamic Programming for Spanning Tree Problems

Luigi Di Puglia Pugliese[a] * Francesca Guerriero[a] José Luis Santos[b]

[a]Department of Electronics, Computer Science and Systems, University of Calabria, Rende (CS), 87036, Italy

[b]Department of Mathematics, University of Coimbra, Coimbra, 3001-454, Portugal

Abstract

The aim of this paper is to provide a dynamic programming formulation for the spanning tree problem ($\mathcal{STP}$). We show that the proposed formulation allows several instances of the classical $\mathcal{STP}$ to be addressed. The spanning tree structure is modelled with states and transition between states, defining a state-space. Several properties are shown and optimality conditions are given. Once the fundamentals of the proposed formulation are shown, the multi-objective spanning tree problem ($\mathcal{MOSTP}$) is addressed. This problem arises in both telecommunications and transportation fields. In addition, the growing in both customers demand and social environment impose that more than one criterion have to be optimized. The scientific literature provides several works that focus on a specialized instance of the considered problem, that is the bi-objective version in which only two criteria are taken into account. To the best of our knowledge, no works provide optimal methods to address the $\mathcal{MOSTP}$ with an arbitrary number l of objective functions. In this paper we extend the proposed dynamic programming formulation to model and solve the $\mathcal{MOSTP}$ with $l \geq 3$ criteria.

Keywords: Dynamic programming; Spanning tree problems; Multiple objective programming; Pareto front.

1 Introduction

The minimum spannning tree problem ($\mathcal{STP}$) aims at finding the shortest undirected paths from a pre-determined source node to each non-source node in a given graph. Many solution approaches to address the $\mathcal{STP}$ were proposed in the scientific literature. For a detailed description and a computational study of these methods, the reader is referred to [1].

When two criteria have to be optimized, the $\mathcal{STP}$ is referred to as bi-objective spanning tree problem ($\mathcal{BOSTP}$). In this case, the optimality of the solutions refers to the concept of dominace. If a tree is not dominated by any other, then it is efficient and the associated cost vector function is said to be Pareto-optimal. They can be considered as best solutions in the sense that no improvement in any criterion is possible without sacrificing at least one of the other criteria. Both exact and approximate solution approaches were developed to solve the $\mathcal{BOSTP}$. The exact methods aim at finding the set of all Pareto-optimal solutions, while the latter provide a sub-set of them or an approximation of the convex hull. Among approximation procedures, we cite the works of Hamacher and Ruhe [4], Knowles and Corne [5]

*Corresponding Author, e-mail: ldipuglia@deis.unical.it

and Zhou and Gen [9]. To the best of our knowledge, only three papers provide exact solution approaches for the $\mathcal{BOSTP}$. Two of these are based on a two-phases procedure (e.g., see [6,8]). In the first step, all the supported non-dominated solutions are determined. In the second step, the objective space within two distinct supported non-dominated solutions (*triangle*) is explored with the aim of finding non-supported non-dominated solutions if any exist. The second phase is run for each consecutive pair of supported non-dominated solutions determined in the first phase. The works of Ramos et al. [6] and Steiner and Radzik [8] mainly differ in the way the Pareto-optimal solutions in the triangles are computed: in [6] these solutions are determined by a branch-and-bound strategy, discarding any node such that an ad-hoc bounding point (that seems to be dominated by the ideal point of the corresponding sub-problem) falls outside the triangles; in [8] the non-supported non-dominated solutions are computed applying a k-best algorithm for the single-objective version of the problem. The k spanning trees are ranked in the ascent direction of the line connecting the two considered supported non-dominated solutions. Steiner and Radzik [8] compared their two-phases method with that proposed in [6] and they concluded that the former significantly outperforms the latter.

The third exact solution approach proposed by Sourd et al. [7] is based on a general branch-and-bound scheme, developed to solve multiple objective optimization problem, that the authors here specialized to the $\mathcal{BOSTP}$. Very recently, Climaco et al. in [3] addressed the minimum cost/minimum label spanning tree problem, by developping path ranking based solution approaches. The multi-objective spanning tree problem ($\mathcal{MOSTP}$) refers to the $\mathcal{STP}$ with more than two objective functions. To the best of our knowledge, no exact procedures have been yet developed to solve the $\mathcal{MOSTP}$.

The main contribution of this paper is to give a general framework to optimally solve instances of the $\mathcal{STP}$. In this paper we focus on the $\mathcal{MOSTP}$ but the proposed approach is likely to address instances like the constrained $\mathcal{STP}$, the k spanning tree problem ($k\mathcal{STP}$), the Steiner tree problem, the constrained counterpart and the k Steiner tree problem. These aspects are better explain in the following. In addition, computational results for the $\mathcal{MOSTP}$ with a number of objective functions greater than 2 are shown for the first time.

The rest of the paper is organized as follows. In Section 2 basic concepts and some notations are given. The general dynamic programming formulation is given in Section 3. The framework described in Section 3 is applied to the $\mathcal{MOSTP}$ in Section 4 along with hints on how to extend the general dynamic programming to different instances of the $\mathcal{STP}$. The developed algorithm for the $\mathcal{MOSTP}$ is tested on a meaningfull number of test problems. The results of the computational experiments are reported in Section 5. Conclusions and final remarks are given in Section 6.

2 Preliminaries and Notations

Let $\mathcal{G}(\mathcal{N},\mathcal{E})$ be a graph where $\mathcal{N}$ is the set of n nodes and $\mathcal{E}$ is the set containig m edges. A tree T in $\mathcal{G}$ is a connected graph $\mathcal{G}_T$ defined by the set of nodes $\mathcal{N}$ and a sub-set of edges $\mathcal{E}_T$ such that $|\mathcal{E}_T| = |\mathcal{N}| - 1$. With each edge $(i,j) \in \mathcal{E}$ is associated a scalar $w_{ij} \geq 0$. Let $\mathcal{T}(\mathcal{G})$ be the set of all spanning trees in graph $\mathcal{G}$. Each tree $T \in \mathcal{T}(\mathcal{G})$ is evaluated by $f(T) = \sum_{(i,j)\in\mathcal{E}_T} w_{ij}$. Let $X_q^{(h)}$ be a sub-set of $\mathcal{N}$ containing q nodes, that is $X_q^{(h)} = \{i_r : r = 1,\ldots,q, q \leq n\}$. The index h means that there exist more than one sub-set with q

nodes. We define as sub-tree $sT_q^{(h)}$ a connected graph with the set of nodes $X_q^{(h)}$ and with the set of edges $\mathcal{E}_q^{(h)}$, where $|\mathcal{E}_q^{(h)}| = |X_q^{(h)}| - 1$.

3 Dynamic Programming based Framework

We denote with $\mathcal{S} = (S, \Gamma)$ the state-space where S is the set of states and Γ is the set of controls. With each state $S_q^{(h)} = (X_q^{(h)}, E_q^{(h)}) \in S$ is associated the sub-tree $sT_q^{(h)}$. There exists a control between two distinct states $S_q^{(h)}$ and $S_{\bar{q}}^{(k)}$ only if it is possible to build a sub-tree $sT_{\bar{q}}^{(k)}$ from the sub-tree $sT_q^{(h)}$ by adding an edge to $\mathcal{E}_q^{(h)}$. Under this respect, it is obvious that controls of the type $(S_q^{(h)}, S_q^{(\bar{h})})$ do not exist. If we suppose to have the control $(S_q^{(h)}, S_q^{(\bar{h})})$, then $\mathcal{E}_q^{(\bar{h})} = \mathcal{E}_q^{(h)} \cup \{(i,j)\}$. Since sub-trees $sT_q^{(h)}$ and $sT_q^{(\bar{h})}$ are composed by the same number of nodes, that is q, the introduction of the arc (i,j) does not imply the introduction of a new node in $X_q^{(\bar{h})}$. This means that $i, j \in X_q^{(h)}$, thus $sT_q^{(\bar{h})}$ is not a sub-tree. It follows that there exists a control $\gamma = (S_q^{(h)}, S_{\bar{q}}^{(k)})$ for each pair of states only if $|X_{\bar{q}}^{(k)}| > |X_q^{(h)}|$. In addition, since $|\mathcal{E}_{\bar{q}}^{(k)}| = |\mathcal{E}_q^{(h)}| + 1$, it follows that $|X_{\bar{q}}^{(k)}| = |X_q^{(h)}| + 1$. In other words, there are controls between pairs of states $S_q^{(h)}$ and $S_{\bar{q}}^{(k)}$, with $\bar{q} = q + 1$. These assumption permits us to consider only feasible transitions. We define as y_ϕ a feasible sequence of transitions, that is, $y_\phi = \{(S_1^{(h)}, S_2^{(k)}), \ldots, (S_{n-1}^{(\bar{h})}, S_n^{(\bar{k})})\}$ and the corresponding sequence of states $y_\phi = \{S_1^{(h)}, \ldots, S_n^{(\bar{k})}\}$. Let Φ be the set containing all feasible sequences. Since with each state is associate a sub-tree, each sequence $y_\phi, \phi = 1, \ldots, |\Phi|$ is associated a tree of $\mathcal{G}(\mathcal{N}, \mathcal{E})$. This implies that $|\Phi| \geq |\mathcal{T}(\mathcal{G})|$.

We assume that a feasible transition from state $S_q^{(h)}$ to state $S_{q+1}^{(k)}$ involves a cost $w(S_q^{(h)}, S_{q+1}^{(k)})$. The objective is to determine a sequence of transitions $y_\phi = \{(S_1^{(h)}, S_2^{(k)}), \ldots, (S_{n-1}^{(\bar{h})}, S_n^{(\bar{k})})\}$ and the corresponding sequence of states $y_\phi = \{S_1^{(h)}, \ldots, S_n^{(\bar{k})}\}$ such that the total cost

$$f(y_\phi) = \sum_{q=1}^{n-1} w(S_q^{(h)}, S_{q+1}^{(k)}) \tag{1}$$

is minimized. Let $T_n^{(k)}$ be the tree associated with the sequence $y_\phi = \{S_1^{(h)}, \ldots, S_n^{(k)}\}$. Tree $T_n^{(k)*} \equiv T_\phi^*$ and the related sequence y_ϕ^* is optimal if and only if $f(y_\phi^*) \leq f(y_{\bar{\phi}})$, $\forall \bar{\phi} = 1, \ldots, |\Phi|, \bar{\phi} \neq \phi$. Since the transition from state $S_q^{(h)}$ to state $S_{q+1}^{(k)}$ corresponds to the insertion of a new edge $(i,j) \in \mathcal{E}$ in the sub-tree associated with the state $S_q^{(h)}$, the cost associated with the transition is equal to the cost associated with the added edge (i,j), that is, $w(S_q^{(h)}, S_{q+1}^{(k)}) = w_{ij}$.

We define as $y_q^{(k)}$ a sub-sequence of states. With $y_q^{(k)}$ is associated the sub-tree $sT_q^{(k)}$. The cost of the sub-sequence $y_q^{(k)}$ is given by $f(y_q^{(k)}) = \sum_{\delta=1}^{q-1} w(S_\delta^{(h)}, S_{\delta+1}^{(k)})$ that is equivalent to $f(sT_q^{(k)}) = \sum_{(i,j) \in \mathcal{E}_q^{(k)}} w_{ij}$.

It is possible to give a definition of optimality for sub-sequences in order to guarantee the optimality of each complete transition sequence. In what follows we give the definition of dominance and equivalence among states that are usuful to state the optimality conditions.

Definition 1. *(Dominance) Let $S_q^{(h)}$ and $S_q^{(\bar{h})}$ be two states belonging to $\mathcal{S}$. We say that $S_q^{(h)}$ dominates $S_q^{(\bar{h})}$ if $f(sT_q^{(h)}) < f(sT_q^{(\bar{h})})$.*

Definition 2. *(Equivalence) Two sub-trees $sT_q^{(k)}$ and $sT_q^{(\bar{k})}$ and the related states $S_q^{(k)}$ and $S_q^{(\bar{k})}$ are said to be equivalent if and only if the corresponding sub-sets $X_q^{(k)}$ and $X_q^{(\bar{k})}$ share the same nodes.*

From definitions 1 and 2, Theorem 1 follows.

Theorem 1. *(Optimality conditions) Let $S_q^{(h)}$ and $S_q^{(\bar{h})}$ be two equivalent states belonging to $\mathcal{S}$. If $S_q^{(h)}$ is dominated by $S_q^{(\bar{h})}$, then all the sub-trees $sT_{q+1}^{(k)}$ built from $sT_q^{(h)}$ are dominated by at least one sub-tree $sT_{q+1}^{(\bar{k})}$ built from $sT_q^{(\bar{h})}$.*

Proof. Let $\mathcal{E}(sT_q^{(h)}) \subseteq \mathcal{E} - \mathcal{E}_q^{(h)} = \{(i^1, j^1), (i^2, j^2), \ldots, (i^\Lambda, j^\Lambda)\}$ be the sub-set of edges that can be used to built $|\mathcal{E}(sT_q^{(h)})|$ sub-trees $sT_{q+1}^{(k)}$ from $sT_q^{(h)}$. Set $\mathcal{E}(sT_q^{(h)})$ contains all edges $(i,j) \in \mathcal{E}$ such that $i \in X_q^{(h)}$ and $j \notin X_q^{(h)}$. It is evident that if two distinct sub-trees $sT_q^{(h)}$ and $sT_q^{(\bar{h})}$ are equivalent, then $\mathcal{E}(sT_q^{(h)}) \equiv \mathcal{E}(sT_q^{(\bar{h})})$. For each $(i^\lambda, j^\lambda) \in \mathcal{E}(sT_q^{(h)})$, the sub-tree $sT_{q+1}^{(k)}(\lambda) = (X_q^{(h)} \cup \{j^\lambda\}, \mathcal{E}_q^{(h)} \cup \{(i^\lambda, j^\lambda)\})$ is built. The corresponding cost is equal to $f(sT_q^{(h)}) + w_{i^\lambda j^\lambda}$. When an equivalent sub-tree $sT_q^{(\bar{h})}$ is considered, since $w_{ij} \geq 0, \forall\ (i,j) \in \mathcal{E}$, if $f(sT_q^{(h)}) \geq f(sT_q^{(\bar{h})})$, then $f(sT_{q+1}^{(k)}(\lambda)) \geq f(sT_{q+1}^{(\bar{k})}(\lambda)), \lambda = 1, \ldots, \Lambda$. This concludes the proof. □

Corollary 1. *The optimal sequence y^* is composed by optimal sub-sequences.*

Proof. From Theorem 1 we know that each state, that is dominated, has not the potential to provide the optimal solution. Thus, a sequence $y^* = \{S_1^{(h)}, \ldots, S_n^{(k)}\}$ is composed by state $S_q^{(h)}$ such that $f(sT_q^{(h)}) < f(sT_q^{(\bar{h})})$, $\forall S_q^{(\bar{h})}$ equivalent to $S_q^{(h)}, q = 1, \ldots, n$. As a consequence, each sub-sequence $y_q^{(h)*}$ is optimal, that is, $f(y_q^{(h)*}) < f(y_q^{(\bar{h})}), q = 1, \ldots, n$. □

Observation 1. *The state-space $\mathcal{S}$ is a layered directed acyclic network.*

Proof. $\mathcal{S}$ can be viewed as composed by n layer. A layer L_q contains all states $S_q^{(h)}$, $h = 1, \ldots, H$ and $q = 1, \ldots, n$. In the layer L_1 there exists the initial state, that is $S_1^{(1)} = (\{i_1\}, \emptyset)$, where i_1 is the root node, whereas layer L_n contains the states associated with each tree in the graph. In addition, since controls between states belonging to the same layer do not exist and each control starting from a layer can end only in the succesive layer, no cycles are present in $\mathcal{S}$. □

4 Application of Dynamic Programming Formulation

In this Section we show the application of the dynamic programming formulation defined in Section 3. In Section 4.1 we propose a new search algorithm in the state-space $\mathcal{S}$ for solving the $\mathcal{MOSTP}$, whereas, in Section 4.2 we give some hints on how use the concepts exposed in Section 3 for modelling and solving several instances of the $\mathcal{STP}$ including the Steiner tree problem ($\mathcal{StTP}$).

4.1 Solving the $\mathcal{MOSTP}$

Let T_ϕ be the spanning tree defined on the connected graph $\mathcal{G}_{T_\phi}(\mathcal{N}, \mathcal{E}_{T_\phi})$. With each edge $(i,j) \in \mathcal{E}$ is associated a weight vector $W_{ij} \in \mathbb{R}^p_+$ containing information about the consumption of weights $w^l_{ij}, l = 1, \ldots, p$ along the edge (i,j). Let $F(T_\phi) = (f^1(T_\phi), \ldots, f^p(T_\phi)) \in \mathbb{R}^p_+$ be the vector cost function associated with the spanning tree T_ϕ. For each weight l, function $f^l(T_\phi), l = 1, \ldots, p$ is defined as $f^l(T_\phi) = \sum_{(i,j)\in T_\phi} w^l_{ij}, \forall l = 1, \ldots, p$. The $\mathcal{MOSTP}$ can be mathematically defined as follows:

$$\min_{T \in \mathcal{T}(\mathcal{G})} F(T) \tag{2}$$

It is worth observing that the optimality condition for the multiple objective programming is generalized in the concept of Pareto efficiency. In what follows, we give the definitions of both dominance and efficiency.

Definition 3. *Let T_ϕ and $T_{\bar{\phi}}$ be two different trees belonging to $\mathcal{T}(\mathcal{G})$. If $f^l(T_{\bar{\phi}}) \geq f^l(T_\phi), \forall\, l = 1, \ldots, p$ and at least one inequality is strictly, then tree T_ϕ dominates tree $T_{\bar{\phi}}$.*

Definition 4. *A spanning tree $T_\phi \in \mathcal{T}(\mathcal{G})$ is said to be an efficient solution if it is not dominated by any $T_{\bar{\phi}} \in \mathcal{T}(\mathcal{G}), T_\phi \neq T_{\bar{\phi}}$.*

Let $\mathcal{T}$ be the set of all efficient solutions. By the definitions introduced above, it follows that $F(T), \forall\, T \in \mathcal{T}$, is a point belonging to the Pareto front, that is, T is a Pareto-optimal solution. Let $\mathcal{F}$ be the set of all Pareto-optimal spanning trees, the developed methods aim at determining the entire set $\mathcal{F} \subseteq \mathcal{T}$, that is, the Pareto front has to be obtained.

From Theorem 1 it is possible to derive dominance relation between sub-trees.

Definition 5. *Let $sT^{(h)}_q$ and $sT^{(\bar{h})}_q$ be two equivalent sub-trees. If $f^l(sT^{(h)}_q) \geq f^l(sT^{(\bar{h})}_q), \forall\, l = 1, \ldots, p$ and at least one inequality is strictly, then $sT^{(\bar{h})}_q$ dominates $sT^{(h)}_q$.*

This results allows us to avoid considering dominated sub-trees. Applying Definition 5 and considering Observation 1, it follows that each layer L_q contains only non-dominated states. Based on these considerations, we can draw a dynamic programming based algorithm to optimally solve the $\mathcal{MOSTP}$. The steps of the proposed solution approach are formally stated in Algorithm 1.

Lemma 1. *The complexity of the proposed Algorithm 1 is $\mathcal{O}\left(\prod_{q=1}^{n-1}[q \times (n-q)]\right)$.*

Proof. At the first level L_1 there is only one state that corresponds to the root node. From this state it is possible to construct exactly $n-1$ sub-trees in the worst case. Thus, layer L_2 contains exactly $n-1$ states. From each state $S^{(h)}_2 \in L_2$ it is possible to construct a sub-tree for each arc $(i,j) \in \mathcal{E}$ such that $i \in X^{(h)}_2$ and $j \in \mathcal{N} \setminus X^{(h)}_2$, that is, $|X^{(h)}_2| \times (n - |X^{(h)}_2|)$. The layer L_3 contains $|L_2| \times [2 \times (n-2)]$ and so on. Thus, the worst case complexity is $\mathcal{O}\left(\prod_{q=1}^{n-1}[q \times (n-q)]\right)$. □

Algorithm 1 *DP*

Step 0 *(Initialization)*
$X_1^1 = \{1\}$; $S_1^1 = (\{1\}, \emptyset)$; $L_1 = \{S_1^1\}$.
Step 1 *(State-space exploration)*
for q=1,. . . ,n-1 **do**
  for all $S_q^{(h)} \in L_q$ **do**
    for all $i \in X_q^{(h)}$ **do**
      for all $(i,j) \in \mathcal{E} : i \in X_q^{(h)}$ AND $j \notin X_q^{(h)}$ **do**
        Set $\bar{X}_{q+1} = X_q^{(h)} \cup \{j\}$.
        Set $\bar{S}_{q+1} = (\bar{X}_{q+1}, \mathcal{E}_q^h \cup \{(i,j)\})$.
        if $s\bar{T}_{q+1}$ is not dominated by any equivalent $sT_{q+1}^{(k)} \in L_{q+1}$ **then**
          Add $s\bar{T}_{q+1}$ to L_{q+1}.
          Remove from L_{q+1} all equivalent sub-trees $sT_{q+1}^{(k)}$ that are dominated by $s\bar{T}_{q+1}$.
        end if
      end for
    end for
  end for
end for
Step 2 *(Exit step)*
L_n contains all Pareto-optimal spanning trees.
Return L_n.

4.2 Other Extentions

The dynamic programming formulation described in Section 3 can be used to model and solve several instances of the classical $\mathcal{STP}$. In particular, it is possible to address the $k\mathcal{STP}$ keeping, for each layer, the first k equivalent sub-trees. The constrained $\mathcal{STP}$ ($\mathcal{CSTP}$) in which an upper bound on the consumption of resources along the tree is imposed, can be easily solved. In particular, it is possible to treat the $\mathcal{CSTP}$ as a $\mathcal{MOSTP}$. The states for which the total amount of resources exceed the upper bounds can be fathomed. The problem of determining the $\mathcal{StTP}$ can be modelled by the dynamic programming formulation proposed in this paper. In this case, the optimal sequence of transitions is that one with minimum cost and containing the minimum number of states such that the set of nodes associated with the last state contains all the Steiner nodes. Under this assumption, all instances of the $\mathcal{StTP}$, that is, the $\mathcal{CStTP}$ and the $k\mathcal{StTP}$ can be addressed as for the $\mathcal{CSTP}$ and the $k\mathcal{STP}$ instances.

5 Computational Experiments

In this Section we describe the computational experiments carried out to test the behaviour of the defined solution approach for the $\mathcal{MOSTP}$. Algorithm 1 is coded in Java language and the tests are performed on Intel core i7 CPU M620 4GB RAM machine. The tests are conducted on instances based on random networks generated by the spacyc generator [2]. The considered networks have a number n of nodes from 5 to 12 and a number of arcs $m = n \times d$, with $d = 5, 10, 15, 20$. For each networks with n nodes and m arcs, we have generated 5 test problems by setting different values for the seed to generate the arc costs. It is worth observing that spacyc code generates parallel arcs. In order to leave only one arc for each set of parallel arcs, a preprocessing in applied for each test problems. After the preprocessing, the number of arcs is reduced and the network does not present parallel arcs. The number of arcs m, after the preprocessing phase, is shown in Table 1.

In addition, we have considered 3 groups of test problems: group 1 contains instances

	n							
d	5	6	7	8	9	10	11	12
5	8.60	13.80	18.00	22.60	26.60	33.80	34.60	39.20
10	9.60	15.00	20.00	27.60	32.80	42.40	49.40	57.40
15	10.00	15.00	20.40	27.80	34.40	43.80	51.80	62.00
20	10.00	15.00	21.00	28.00	35.80	45.00	53.40	64.80

Table 1: Number of arcs m for each network after the preprocessing.

with 3 criteria, instances with 4 criteria belong to group 2, whereas in group 3 instances with 5 criteria are included. Thus we consider a total of 480 instances. In what follows, we refer to R1 as the set of networks with 5 nodes, with R2 to those with 6 nodes and so on. The collected computational results are shown in Table 2, where each line reports the average results on 20 instances.

	3 criteria			4 criteria			5 criteria		
Tests	#states	#PO	time	#states	#PO	time	#states	#PO	time
R1	60.20	16.65	0.00	81.25	29.30	0.00	109.35	46.90	0.00
R2	224.37	42.89	0.00	414.68	121.79	0.02	675.95	245.79	0.04
R3	781.80	92.10	0.08	1491.00	269.65	0.20	2900.45	753.05	0.79
R4	2165.65	135.15	0.39	6069.60	661.60	2.96	13570.45	2296.20	16.43
R5	6950.60	256.80	4.43	20086.25	1370.95	36.66	61034.50	6667.90	709.52
R6	23185.05	458.20	85.44	83652.80	3629.15	1851.20	$178703.00^{(9)}$	$12326.44^{(9)}$	$9627.69^{(9)}$
R7	57142.80	549.10	1133.95	$122330.16^{(6)}$	$2665.67^{(6)}$	$5287.97^{(6)}$			
R8	166608.90	910.05	13835.09						
AVG 1	32139.92	307.62	1882.42	33446.53	1249.73	1025.57	42832.28	3722.71	1725.74
AVG 2	2036.52	108.72	0.98	5628.56	490.66	7.97	15658.14	2001.97	145.35

Table 2: Computational results collected on test problems with 3, 4 and 5 criteria. Under columns #states we report the number of states explored by Algorithm 1, the number of Pareto-optimal trees is shown under columns #PO. The computational cost in terms of execution time, given in seconds, is reported under columns time.

In what follows, we discuss about the computational results on the average case. In particular, we refer to the values reported in Table 2. Of course, it is interesting to evaluate the behaviour of the proposed dynamic programming based algorithm considering the number of arcs m and consequently, the value of d. In Figure 1, the number of states and the number of Pareto-optimal trees are plotted with respect to the number of nodes n and, for each n, the results for each value of d are reported. We remark that the parameter d defines the number of arcs. However, the values of m are those reported in Table 1.

From Figure 1, it is observed that the higher the parameter d the higher the number of states and the number of Pareto-optimal trees for values of d from 5 to 15. A slightly decreases in both the number of states and the number of Pareto-optimal trees is observed for $d = 20$.

Table 2 shows the average results on all the solved test problems for each group (see column AVG 1), that is, on test problems R1 - R8, R1 - R7 and R1 - R6 for group 3, 4 and 5, respectively. In order to give a comparison among the groups, we report the average results on test problems that are solved by all groups under column AVG 2, that is, the average results on test problems R1 - R5 for each group. The empty entries mean that the corresponding test problems are not solved within the imposed time limit of 4 hours (14400 seconds). In addition, the superscript indicate the number of solved instances.

From Table 2 it is observed the expected trend. Indeed, the higher the number of criteria the higher the number of Pareto-optimal spanning trees. This behaviour is observed for all the considered test problems. In addition, as drawn in Figure 2, for each group, a remarkable growing of the number of both the explored states and the Pareto-optimal trees with respect

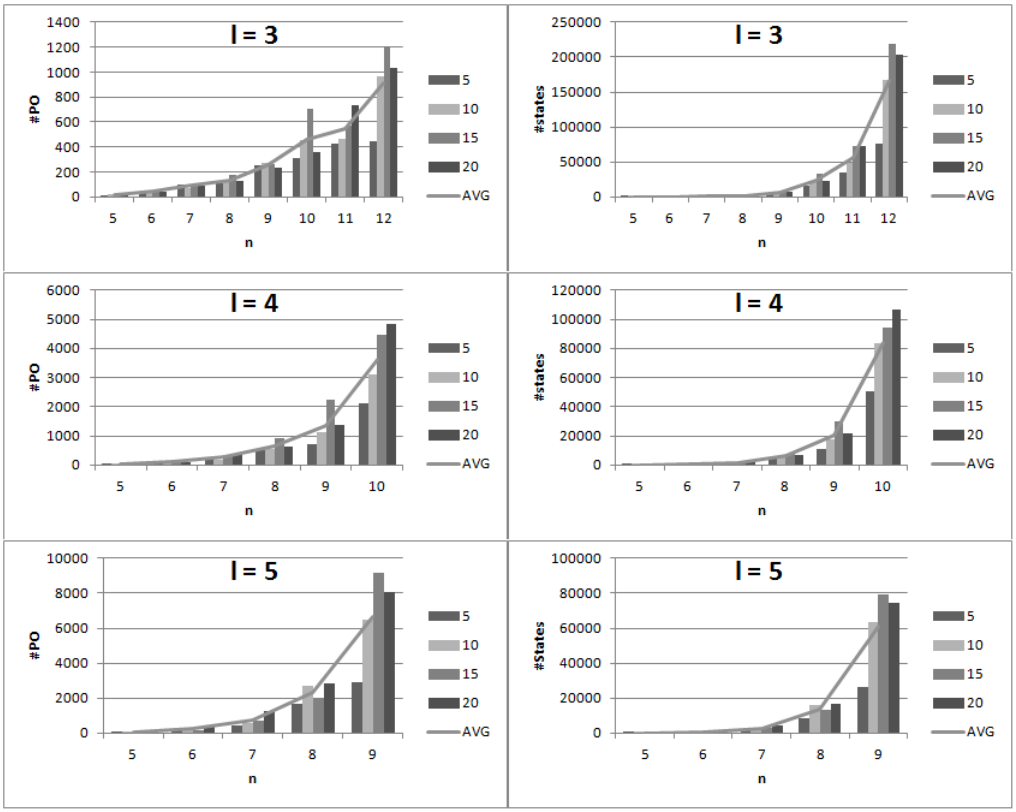

Figure 1: Computational results plotted with respect to the values of n and d.

to the increase of the dimensions of the networks is observed.

The computational results collected in Table 2 underline that it is necessary to explore an high number of states. In particular, the number of states is, on average, 49.77, 14.92 and 6.42 times higher than the number of Pareto-optimal trees for group 1, 2 and 3, respectively. It is worth observing that for groups 2 and 3 we also consider the solved instances of networks R7 and R6, respectively. The differences among the groups is related to the fact that, for group 1, networks with an higher number of nodes (R1 - R8) are solved than those solved for group 2 (R1 - R7) and group 3 (R1 - R6). Indeed, the ratio between #states and #PO increases with the number of nodes but decreases with the number of criteria in the test problems. Regarding the computational cost, the time spent for solving the test problems belonging to group 2 and 3 is, on average (see AVG 2), 8.12 and 148.15 times higher than the computational cost for solving those in group 1. This behaviour can be justified by considering the number of explored states. Indeed, the states explored by Algorithm 1 to solve the test problems in group 2 and 3 is, on average, 2.76 and 7.69 times higher than those explored for the test problems belonging to group 1. However, the increase in terms of computational cost is more higher than the growing in terms of explored states. This behaviour is due to the fact that the time for dominance check has to be taken into account.

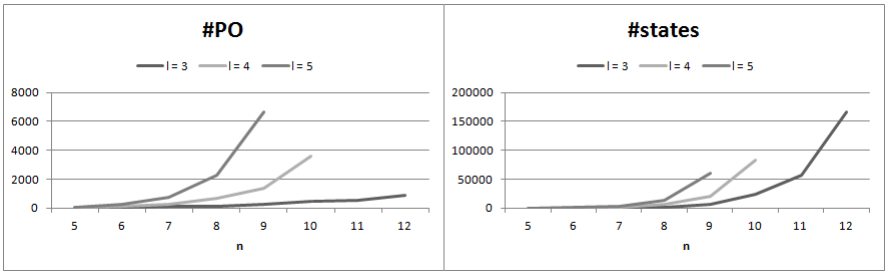

Figure 2: Number of Pareto-optimal trees and number of states plotted with respect to the number of nodes n, for each group of instances.

6 Conclusions

This paper is divided into two parts. In the first one, a new formulation for the spanning tree problems is given. The problem is viewed under a dynamic programming framework. Indeed, the feasible region is formulated using the concepts of states and transitions among states, defining a state-space. We conduct an in depth study about the characteristics of the defined state-space. We focus on its structure and we derive useful properties. Optimal conditions for the defined state-space are given. In the second part, we define an optimal algorithm dealing with the multi-objective spanning tree problem with an arbitrary number of objective functions. The defined algorithm is based on the properties and optimal conditions given in the first part. In addition, computational study for instances with more than 2 objective functions is provided for the first time. We show that the new formulation can be used to model and solve a variety of instances of the spanning tree problem including the Steiner tree. In particular, the properties and the optimal conditions of the defined state-space can be extended to solve the k spanning tree, the costrained spanning tree, the k Steiner tree and the constrained Steiner tree probelms. These problems are the subjects of current investigation.

References

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

[2] B.V. Cherkassky, A.V. Goldberg, and T. Radzik. Shortest paths algorithms: Theory and experimental evaluation. *Mathematical Programming*, 73(2):129–174, 1996.

[3] J.C.N. Clmaco, M. Eugnia Captivo, and M.M.B. Pascoal. On the bicriterion minimal cost/minimal label spanning tree problem. *European Journal of Operational Research*, 2009. doi: 10.1016/j.ejor.2009.10.013.

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

[5] J.D. Knowles and D.W. Corne. A comparison of encodings and algorithms for multi-objective spanning tree problems. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 544–551, 2001.

[6] R.M. Ramos, S. Alonso, J. Sicilia, and Gonzales C. The problem of the optimal biobjective spanning tree. *European Journal of Operational Research*, 111:617–628, 1998.

[7] F. Sourd, O. Spanjaard, and P. Perny. Multi-objective branch-and-bound. application to the bi-objective spanning tree problem. In *MOPGP06: 7th Int. Conf. on Multi-Objective Programming and Goal Programming*, 2006.

[8] S. Steiner and T. Radzik. Computing all efficient solutions of the biobjective minimum spanning tree problem. *Computers & Operations Research*, 35:198–211, 2008.

[9] G. Zhou and M. Gen. Genetic algorithm approach on multi-criteria minimum spanning tree problem. *European Journal of Operational Research*, 114:141–152, 1999.