

Instituto de Engenharia de Sistemas e Computadores de Coimbra  
Institute of Systems Engineering and Computers  
INESC – Coimbra

João Clímaco and Marta Pascoal

**Finding non-dominated shortest pairs of disjoint simple paths - a revised version**

No. 10

2008

ISSN: 1645-2631

Instituto de Engenharia de Sistemas e Computadores de Coimbra  
INESC – Coimbra  
Rua Antero de Quental, 199; 3000 - 033 Coimbra; Portugal  
[www.inescc.pt](http://www.inescc.pt)

# FINDING NON-DOMINATED SHORTEST PAIRS OF DISJOINT SIMPLE PATHS - A REVISED VERSION

JOÃO C. N. CLÍMACO<sup>(1,2)</sup>, and MARTA M. B. PASCOAL<sup>(2,3)</sup>

<sup>(1)</sup> Faculty of Economics, University of Coimbra, Avenida Dias da Silva, 165, 3004-512 Coimbra, Portugal

<sup>(2)</sup> Instituto de Engenharia de Sistemas e Computadores – Coimbra, Rua Antero de Quental, 199, 3000-033 Coimbra, Portugal

E-mail: *jclimaco@inescc.pt*

<sup>(3)</sup> Department of Mathematics, University of Coimbra, Apartado 3008, 3001-454 Coimbra, Portugal

E-mail: *marta@mat.uc.pt*

August 2008

---

**Abstract:** In this paper we introduce a method to compute non-dominated bicriteria shortest pairs, each including two disjoint simple paths. The method is based on an algorithm for ranking pairs of disjoint simple paths by non-decreasing order of cost, that is an adaptation of a ranking path algorithm applied to a network obtained from the original one after a suitable modification of the topology. Each path in this new network corresponds to a pair of paths in the former one. Computational results are presented and analysed for randomly generated networks.

**Keywords:** Disjoint paths, shortest simple paths, multicriteria, ranking algorithms.

---

## 1 Introduction

The shortest path problem is one of the most popular in operations research. It aims to compute a route in a network that can be used to send data or goods from one place to another, minimizing an objective function that can represent a cost or time for using that route. Some of the, many, variants of the shortest path problem intend to determine more than one route, either for having alternatives to the best route in case of failure, or to use those routes simultaneously to spread the information transmitted at a time. In order to cope with this latter problem disjoint paths, that is paths with no intermediate nodes in common, that do not share network resources, with minimum cost are considered. Given  $k \in \mathbb{N}$ , the determination of  $k$  disjoint simple paths has been treated by Suurballe, Suurballe and Tarjan, and Bhandari [6, 7, 1]. These approaches consist of formulating the problem as a minimum cost flow problem and propose the application of a labeling algorithm, changing the given network.

We present a modification of the topology of a graph that changes the determination of a set of disjoint paths into listing path by order of a cost function. This work was motivated by an application to a multicriteria routing model for MPLS (Multiprotocol Label Switching) Networks with traffic splitting in which disjoint pairs of simple paths (DPSP) are used to spread the traffic. This model involves two objective functions to be minimized, namely the cost of using the two paths, given by the sum of their arc costs, which measures the load balancing cost, and the number of arcs in the paths [2, 3]. To deal with this traffic splitting problem a method for listing shortest disjoint pairs of simple

paths (SDPSP) was developed, and since we needed to minimize two objective functions we looked only for the non-dominated pairs. Here we consider two additive cost functions. The first purpose of the paper is to introduce a way to determine DPSP, producing a method able also to list these paths by non-decreasing order of cost. Finally that ranking method is combined with a dominance test thus leading to a way to generate all the non-dominated solutions when two costs are considered.

The paper has four more sections. Section 2 introduces some notation and the problem addressed. Section 3 proposes an algorithm for listing PSDP by order of cost, after a proper modification in the network topology. Section 4 focuses on a method for finding non-dominated PDSP concerning two objective functions. Finally, tested examples in randomly generated networks are presented and the results are discussed in Section 5.

## 2 Problem definition

Let  $(\mathcal{N}, \mathcal{A})$  be a network with a set  $\mathcal{N}$  of  $n$  nodes and a set  $\mathcal{A}$  of  $m$  arcs. A path  $p$  from  $i \in \mathcal{N}$  to  $j \in \mathcal{N}$  in  $(\mathcal{N}, \mathcal{A})$  is a sequence of the form  $p = \langle i = v_1, v_2, \dots, j = v_{\ell(p)} \rangle$ , where  $(v_k, v_{k+1}) \in \mathcal{A}$ , for any  $k \in \{1, \dots, \ell(p) - 1\}$ . Nodes  $i$  and  $j$  are called initial and terminal nodes of  $p$ , respectively. If  $x$  and  $y$  are nodes of  $p$ , then the subsequence of  $p$  between  $x$  and  $y$  is represented by  $\text{sub}_p(x, y)$ , and called the subpath of  $p$  from  $x$  to  $y$ .

**Definition 1** *A path  $p$  is said to be simple (or loopless) if it has no repeated nodes.*

Let  $\mathcal{P}_{xy}$  denote the set of paths from node  $x$  to node  $y$  in  $(\mathcal{N}, \mathcal{A})$ , and  $\mathcal{P}$  denote the set of paths from a source node  $s$  to a terminal node  $t$  (that is,  $\mathcal{P}_{st} = \mathcal{P}$ ).

**Definition 2** *Two paths  $p, q$  from  $x$  to  $y$  are said to be disjoint if the only nodes they have in common are  $x$  and  $y$ .*

Given  $p \in \mathcal{P}_{xi}$  and  $q \in \mathcal{P}_{iy}$  let  $p \diamond q$  represent the concatenation of those paths, that is,  $p \diamond q$  is the path formed by  $p$  and followed by  $q$ . With each arc  $(i, j)$  of the network two costs are associated:  $c_{xy}^i \in \mathbb{R}$ , with  $i = 1, 2$ . Given any path  $p$  between a pair of nodes in a network we also define two objective functions:

$$c_i(p) = \sum_{(x,y) \in p} c_{xy}^i, \quad i = 1, 2,$$

which we intend to minimise. Our aim is the determination of a DPSP from  $s$  to  $t$  that minimise both objective functions.

## 3 Determination of disjoint pairs of simple paths

The determination of DPSP is discussed in this section. The first part concerns a modification of the given graph, aiming at transforming pairs of simple paths between a pair of nodes into simple paths of the modified network. The second part gives a method for listing DPSP by non-decreasing order of a cost function, providing an algorithm for finding them. This algorithm ranks paths that correspond to pairs of paths in the original network, minimises the number of pairs with common nodes and filters the disjoint pairs.

### 3.1 Network topology modification

Let  $(\mathcal{N}, \mathcal{A})$  be a given network and  $s, t$  be an origin-destination pair of nodes in  $\mathcal{N}$ . Let  $(\mathcal{N}', \mathcal{A}')$  be a new network, such that:

- the former nodes are duplicated:  $\mathcal{N}' = \mathcal{N} \cup \{i' : i \in \mathcal{N}\}$ ,
- the former arcs are duplicated and a new one, linking  $t$  and the new node  $s'$ , is added:  $\mathcal{A}' = \mathcal{A} \cup \{(i', j') : (i, j) \in \mathcal{A}\} \cup \{(t, s')\}$ .

In the new network the initial node  $s$  is maintained and a new terminal node,  $t'$ , is considered. The costs of the original arcs are maintained, while for the new ones we have  $c_{i'j'} = c_{ij}$ , if  $(i, j) \in \mathcal{A}$ , and  $c_{t,s'} = 0$ . Figure 1 shows an example of a network and the correspondent augmented one.

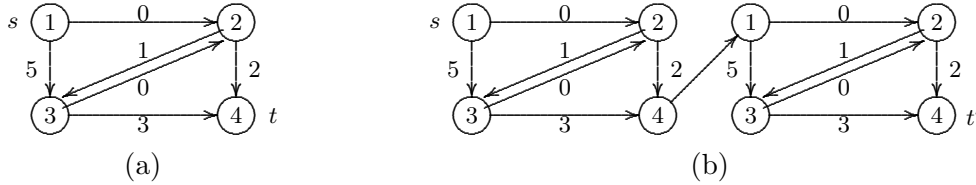


Figure 1: Networks (a)  $(\mathcal{N}, \mathcal{A})$  and (b)  $(\mathcal{N}', \mathcal{A}')$

Each path  $p$  from  $s$  to  $t'$  in  $(\mathcal{N}', \mathcal{A}')$  corresponds to a pair of paths from  $s$  to  $t$  in  $(\mathcal{N}, \mathcal{A})$ , such that

$$p = q \diamond (t, s') \diamond q',$$

where  $q \in \mathcal{P}_{st}$  and  $q' \in \mathcal{P}'_{s't'}$ . Thus, if  $q$  and  $q'$  are simple paths and in addition  $q \cap q' = \emptyset$ , then  $q, q'$  do not share nodes and thus correspond to a DPSP in  $(\mathcal{N}, \mathcal{A})$ . The following definitions intend to simplify the implementation of a method to find the SDPSP.

**Definition 3** A path  $p$  is simple iff for any node  $x \in p$ , if  $y$  is a node of  $p$  previous to  $x$  then  $y \notin \text{sub}_p(x, t')$ .

**Definition 4** Two simple paths  $q_1, q_2 \in \mathcal{P}$  are disjoint iff, for any node  $x \neq s, t$ , if  $x \in q_1$  then  $x \notin q_2$ .

Using a ranking algorithm we can list the simple paths from  $s$  to  $t'$ , and check whether each one corresponds to a DPSP. This allows to filter only the DPSP, ordering them by cost.

**Remark 1** Regarding the memory space demanded to store the modified network  $(\mathcal{N}', \mathcal{A}')$  as defined above, the number of nodes in  $\mathcal{N}'$  is  $2n$  and the number of arcs in  $\mathcal{A}'$  is  $2m + 1$ . However, every new arc besides  $(t, s')$  is a copy of other existent arc, therefore the duplication of such arcs can be avoided, as long as the correspondent arcs in  $\mathcal{A}$  are analysed whenever an arc in  $\mathcal{A}' - \{(t, s')\}$  is considered, therefore only  $m + 1$  arcs need to be stored. It should be noticed that in these case operations should be implemented cautiously if the network is subject to other modifications, as the sets of arcs associated with a node in  $\mathcal{N}$  and its correspondent in  $\mathcal{N}'$  might become different.

### 3.2 Ranking disjoint pairs of simple paths by cost

The ranking of DPSP by non-decreasing order of cost can be made with an adaptation of the algorithm by Martins, Pascoal and Santos [5] for ranking simple paths. This algorithm, which is now briefly reviewed, allows to easily incorporate additional tests on paths and thus reduces the number of candidates that have to be generated.

Let  $X$  be a set that stores simple path candidates to  $p_k$ ,  $k = 1, \dots, K$ . The set  $X$  is initialised with the shortest path from  $s$  to  $t$  in  $(\mathcal{N}, \mathcal{A})$ ,  $p_1$ , and if  $p_1, \dots, p_{k-1}$  have been determined then  $p_k$  is the shortest candidate in  $X$ . When  $p_k$  is selected, and deleted, in  $X$  its nodes are analysed, in order to generate new candidates with a low cost. The shortest deviation of  $p_k$  at one of its nodes is given by the shortest path from that node to  $t$ , after the arc of  $p_k$  that starts at that node being deleted, and the best path from  $s$  to  $t$  that deviates from  $p_k$  at  $v_i$  has the form:

$$p = \text{sub}_{p_k}(s, v_i) \diamond (v_i, j) \diamond \mathcal{T}_t(j),$$

where  $(v_i, j)$  does not belong to any of the candidates computed so far,  $\mathcal{T}_t$  denotes the tree of shortest paths from any node to  $t$ , and  $\mathcal{T}_t(j)$  denotes the path from  $j$  to  $t$  in  $\mathcal{T}_t$ ,  $\pi_j$  stands for  $c(\mathcal{T}_t(j))$ . Node  $v_i$  is called the deviation node of  $p$ , and it will be denoted by  $d_p$ .

The generation of a new candidate by this method depends on the selection of an arc, which can take into account the two constraints considered, namely: the nodes of a path should not be repeated, and the nodes of paths that belong to  $\mathcal{N}$  and to  $\mathcal{N}'$  should be distinct, except  $s, t, s', t'$ . This procedure can produce paths with loops or with common nodes in  $\mathcal{N}$  and  $\mathcal{N}'$  whenever  $\text{sub}_{p_k}(s, v_i)$  and  $\mathcal{T}_{t'}(j)$  share nodes. Therefore, only arcs that start at a node previous to the first loop have to be considered. A sketch of this algorithm is presented in the following, and further details on its original version can be found in [5]. Algorithm 1 refers to function *Disjoint*, that checks if a path  $p$  in  $(\mathcal{N}', \mathcal{A}')$  corresponds to a DPSP in  $(\mathcal{N}, \mathcal{A})$ . In other words this function checks whether the path  $p \cap \mathcal{N}$  and the path that corresponds to  $p \cap \mathcal{N}'$  in the original network share other nodes besides the initial and the terminal.

**Algorithm 1** – *Determination of a shortest pair of disjoint simple paths concerning cost function  $c$*   
 $\mathcal{T}_{t'} \leftarrow$  tree of the shortest paths from  $i \in \mathcal{N}'$  to  $t'$  concerning  $c$   
 $p \leftarrow \mathcal{T}_{t'}(s)$   
**If** ( $p$  is not defined) **Then Stop** /\* There is no pair of disjoint simple paths \*/  
 $\bar{c}_{ij} \leftarrow \pi_j - \pi_i + c_{ij}, \forall (i, j) \in \mathcal{A}'$   
 Represent  $\mathcal{A}'$  in the sorted forward star form concerning  $\bar{c}$   
 $d_p \leftarrow s; X \leftarrow \{p\}; \text{feasible} \leftarrow \text{False}$   
**While** ( $(X \neq \emptyset)$  and (not *feasible*)) **Do**  
    $p \leftarrow$  path in  $X$  such that  $\bar{c}(p)$  is minimum; /\*  $p = \langle s \equiv v_1, v_2, \dots, v_{\ell-1}, v_\ell \equiv t' \rangle$  \*/  
    $X \leftarrow X - \{p\}$   
    $i \leftarrow$  index such that  $v_i = d_p$   
   **While** ( $(v_i \neq t')$  and ( $\text{sub}_p(s, v_i)$  is simple) and *Disjoint*( $\text{sub}_p(s, v_i)$ )) **Do**  
      $l \leftarrow$  index such that  $a_l = (v_i, v_{i+1})$   
     **While** ( $(v_i$  is the tail node of  $a_l$ ) and  
       ( $(a_{l+1}$  forms a loop with  $\text{sub}_p(s, v_i)$ ) or not *Disjoint*( $\text{sub}_p(s, v_i) \diamond a_{l+1}$ )) **Do**  
        $l \leftarrow l + 1$   
     **EndWhile**  
     **If** ( $v_i$  is the tail node of  $a_l$ ) **Then**  
        $v_j \leftarrow$  head node of  $a_l; q \leftarrow \text{sub}_p(s, v_i) \diamond a_l \diamond \mathcal{T}_t(v_j); d_q \leftarrow v_i; X \leftarrow X \cup \{q\}$   
     **EndIf**  
      $v_i \leftarrow v_{i+1}$   
   **EndWhile**  
   **If** ( $(p$  is simple) and *Disjoint*( $p$ )) **Then** *feasible*  $\leftarrow$  True  
**EndWhile**  
**If** (not *feasible*) **Then Stop** /\* There is no pair of disjoint simple paths \*/

Note that the first steps of MPS algorithm are computing  $\mathcal{T}_t$  and replacing arc costs by reduced costs. The shortest tree from any node to the terminal and the new costs are depicted in Figures 2



Figure 2: (a)  $\mathcal{T}_t$  in  $(\mathcal{N}, \mathcal{A})$  and (b)  $(\mathcal{N}, \mathcal{A})$  with reduced costs

and 3 when considering the original network  $(\mathcal{N}, \mathcal{A})$  and the duplicated network  $(\mathcal{N}', \mathcal{A}')$ , both in Figure 1, respectively.

As shown in the pictures, and also easy to prove,

$$\pi_i = \pi_{i'} + \pi_s, \text{ for any } i \in \mathcal{N}.$$

As a consequence it is possible to compute  $\mathcal{T}_t$  before  $(\mathcal{N}, \mathcal{A})$  being duplicated, and to use that information in the modified network with no need to solve a new single source shortest path problem and determine  $\mathcal{T}_{t'}$ . Moreover, if the network arcs are not replicated, as suggested in Remark 1, then for any  $i \in \mathcal{N}$ ,  $(i', j') \in \mathcal{A}'$  and  $\bar{c}_{i'j'} = c_{ij}$ . Similarly,  $(j', i') \in \mathcal{A}'$  and  $\bar{c}_{j'i'} = c_{ji}$ . The only arc that has to be treated differently being  $(t, s')$ , as it has no correspondent in the original network.

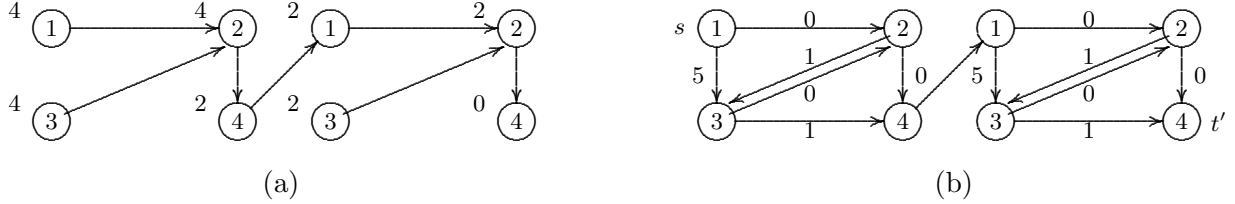


Figure 3: (a)  $\mathcal{T}_{t'}$  in  $(\mathcal{N}', \mathcal{A}')$  and (b)  $(\mathcal{N}', \mathcal{A}')$  with reduced costs

When the original graph is duplicated not only the size of the problem but also the number of solutions that can be obtained is doubled, as equivalent pairs of paths, like  $(p, q')$  and  $(q, p')$ , can be generated. For instance, when applied to the network of Figure 1 Algorithm 1 first determines the shortest path from 1 to 4',  $p = \langle 1, 2, 4, 1', 2', 4' \rangle$ , which corresponds to pair  $(p, q')$ , where  $q = \langle 1, 2, 4 \rangle$  and  $q' = \langle 1', 2', 4' \rangle$ , with cost  $(2, 2)$ . Since node 2' belongs to  $q$  path  $p$  does not lead to disjoint paths in the original graph, but still its nodes from 1 to 1' are scanned, and the following paths are obtained:

Path	Scanned node	Generated path	Costs pair
$q_1$	1	$\langle 1, 3, 2, 4, 1', 2', 4' \rangle$	$(7, 2)$
$q_2$	2	$\langle 1, 2, 3, 4, 1', 2', 4' \rangle$	$(3, 2)$
$q_3$	1'	$\langle 1, 2, 4, 1', 3', 2', 4' \rangle$	$(2, 7)$

and it is easy to see  $q_1$  and  $q_3$  correspond to the same pair of paths.

**Remark 2** Based on this example one can discard equivalent pairs of simple paths, or else derive a pruning technique that prevents one of those paths from being computed. This can be done by allowing only paths  $p$  such that  $c(\text{sub}_p(s, t)) \leq c(\text{sub}_p(s', t'))$  (or instead  $c(\text{sub}_p(s, t)) \geq c(\text{sub}_p(s', t'))$ ), since the symmetric path will still be obtained. Thus, aiming to reduce the number of paths to be stored, the inner **While** loop in Algorithm 1 can be replaced by:

**While** ( $(v_i$  is the tail node of  $a_l$ ) **and**  $\text{EquivalentPair}(\text{sub}_p(s, v_i) \diamond a_{l+1} \diamond \mathcal{T}_t(\text{head node of } a_{l+1}))$ ) **and**  $((a_{l+1}$  forms a loop with  $\text{sub}_p(s, v_i))$  **or not**  $\text{Disjoint}(\text{sub}_p(s, v_i) \diamond a_{l+1}))$ ) **Do**

where *EquivalentPair* stands for a boolean function that given a path  $p$  returns the value **True** iff

$$c(\text{sub}_p(s, t)) > c(\text{sub}_p(s', t')).$$

**Remark 3** Finally, it should also be noticed that to allow ranking DPSP by order of the objective function  $c$  one only has to change the stopping condition of the outer **while** loop in this algorithm. Moreover, if the network  $(\mathcal{N}, \mathcal{A})$  is replicated  $k$  times,  $k \in \mathbb{N}$ , then Algorithm 1 can be applied to determine the shortest set of  $k$  disjoint simple paths.

## 4 Method for finding non-dominated solutions

In the bicriteria problem of finding non-dominated DPSP we intend to find DPSP from  $s$  to  $t$  that minimise the functions  $c_1$  and  $c_2$  in  $\mathcal{P}$ . If the objective functions are conflictuous there is no optimal solution for this problem, therefore we will determine a set of non-dominated solutions, formed by DPSP such that there is no other which improves one objective function without worsening the other.

Given  $(p_i, q_i)$ ,  $i = 1, 2$ , two DPSP from  $s$  to  $t$  in  $(\mathcal{N}, \mathcal{A})$ , we define that pair  $(p_1, q_1)$  dominates pair  $(p_2, q_2)$ , written  $(p_1, q_1)_D(p_2, q_2)$ , if and only if  $c_i(p_1) + c_i(q_1) \leq c_i(p_2) + c_i(q_2)$ ,  $i = 1, 2$ , and at least one of the inequalities is strict.  $(p_1, q_1)$  is non-dominated if there is no DPSP that dominates it. This notions can be extended for simple paths from  $s$  to  $t'$  in the modified network  $(\mathcal{N}', \mathcal{A}')$ .

In 1982 Clímaco and Martins [4] introduced a method to solve the bicriteria shortest path problem, that is adapted here to determine the set of non-dominated DPSP. The idea is listing solutions by non-decreasing order of one of the objective functions (using the method described in Section 3), and add a dominance test to this enumeration in order to filter only the non-dominated solutions. The dominance test results from the observation that if the enumeration is made by non-decreasing order of the first objective function, then the sequence of the second objective function values of non-dominated solutions is non-increasing. Thus, every time a simple path  $p$  in  $(\mathcal{N}', \mathcal{A}')$  is found the correspondent objective function values,  $(c_1(p), c_2(p))$ , have to be compared with the previously obtained. Let  $M_1$  be the greatest value of  $c_1$  determined so far, and let  $m_2$  be the least value of  $c_2$  determined. The two possible situations are:

1.  $c_1(p) = M_1$ 
  - If  $c_2(p) > m_2$  then  $p$  is dominated.
  - If  $c_2(p) < m_2$  then  $p$  dominates the solutions with cost  $(M_1, m_2)$  and  $p$  is a candidate to non-dominated solution. The value  $m_2$  is updated.
  - If  $c_2(p) = m_2$  then  $p$  is a candidate to non-dominated solution.
2.  $c_1(p) > M_1$ 
  - If  $c_2(p) \geq m_2$  then  $p$  is dominated.
  - If  $c_2(p) < m_2$  then the solutions with cost  $(M_1, m_2)$  are non-dominated and  $p$  is a candidate to non-dominated solution. The values  $M_1$  and  $m_2$  are updated.

The method to find the set of non-dominated solutions incorporates this dominance test in a ranking algorithm. Let  $p_i^*$  be a path in  $(\mathcal{N}', \mathcal{A}')$  that corresponds to a SDPSP in  $(\mathcal{N}, \mathcal{A})$  in terms of  $c_i$ ,  $i = 1, 2$ , and let

$$c_i^* = c_i(p_i^*), \quad i = 1, 2, \quad \hat{c}_1 = c_1(p_2^*), \quad \hat{c}_2 = c_2(p_1^*).$$

Assuming the solutions are ranked according to  $c_1$  the first solution,  $p$ , satisfies  $c_1(p) = c_1^*$ , then the values of  $c_1$  increase along the algorithm and all the non-dominated solutions satisfy  $c_1(p) \leq \hat{c}_1$  and

$c_2(p) \geq c_2^*$ . The scheme bellow summarises the method to find non-dominated solutions, using the procedure described in the previous section to rank DPSP by cost.

- Find the shortest simple path from  $s$  to  $t'$  in  $(\mathcal{N}', \mathcal{A}')$  concerning  $c_2$  corresponding to a DPSP in  $(\mathcal{N}, \mathcal{A})$ ,  $p = q \diamond (t, s') \diamond q'$
- $\hat{c}_1 \leftarrow c_1(p)$ ,  $c_2^* \leftarrow c_2(p)$
- Find the shortest simple path from  $s$  to  $t'$  in  $(\mathcal{N}', \mathcal{A}')$  concerning  $c_1$  corresponding to a DPSP in  $(\mathcal{N}, \mathcal{A})$ ,  $p_1 = q \diamond (t, s') \diamond q'$
- $m_2 \leftarrow \hat{c}_2 \leftarrow c_2(p_1)$ ,  $M_1 \leftarrow c_1^* \leftarrow c_1(p_1)$
- $k \leftarrow 1$
- While  $c_1(p_k) \leq \hat{c}_1$  and  $c_2(p_k) \geq c_2^*$  Do
  - $k \leftarrow k + 1$
  - Find the next shortest simple disjoint path from  $s$  to  $t'$  in  $(\mathcal{N}', \mathcal{A}')$  concerning  $c_1$  corresponding to a DPSP in  $(\mathcal{N}, \mathcal{A})$ ,  $p_k = q_k \diamond (t, s') \diamond q'_k$ .
  - Apply the dominance test to  $p_k$  and update  $M_1, m_2$  and  $\mathcal{P}_N$ .

**Algorithm 2** – *Determination of non-dominated shortest pairs of disjoint simple paths*

Construct the modified network  $(\mathcal{N}', \mathcal{A}')$

$p_{c_2}^* \leftarrow$  shortest pair of disjoint simple paths concerning  $c_2$ ;  $\hat{c}_1 \leftarrow c_1(p_{c_2}^*)$

$p_{c_1}^* \leftarrow$  shortest pair of disjoint simple paths concerning  $c_1$ ;  $M_1 \leftarrow c_1(p_{c_1}^*)$ ;  $m_2 \leftarrow c_2(p_{c_1}^*)$

$\mathcal{P}_X \leftarrow \{p_{c_1}^*\}$ ;  $\mathcal{P}_N \leftarrow \emptyset$ ; *continue*  $\leftarrow$  True;  $k \leftarrow 0$

While (*continue*) Do

$k \leftarrow k + 1$

$p_k \leftarrow$   $k$ -th shortest feasible simple path concerning  $c_1$

If ( $c_1(p_k) = M_1$ ) Then

/\* Dominance test \*/

If ( $c_2(p_k) = m_2$ ) Then  $\mathcal{P}_X \leftarrow \mathcal{P}_X \cup \{p_k\}$

Else

If ( $c_2(p_k) < m_2$ ) Then

$\mathcal{P}_X \leftarrow \{p_k\}$ ;  $m_2 \leftarrow c_2(p_k)$

EndIf

EndIf

Else

If ( $c_2(p_k) < m_2$ ) Then

$\mathcal{P}_N \leftarrow \mathcal{P}_N \cup \mathcal{P}_X$ ;  $\mathcal{P}_X \leftarrow \{p_k\}$ ;  $M_1 \leftarrow c_1(p_k)$ ;  $m_2 \leftarrow c_2(p_k)$

If ( $c_1(p_k) > \hat{c}_1$ ) Then *continue*  $\leftarrow$  False

EndIf

EndIf

EndWhile

## 5 Computational tests

The approaches proposed in the previous sections to deal with DPSP were implemented in C language and some experiments were run on a laptop, Core 2 at 1.66 GHz, with 1 Gb of RAM. First we implemented two versions of Algorithm 1 for ranking disjoint pairs of simple paths, the original method, **ORA**, and the modification that reduces the number of repeated pairs of paths, **MRA**, described in



	Min.	Average	Max.		Min.	Average	Max.
(2)	2	11.980	162	(1)	0.000	0.001	0.004
(3)	2	14.600	156	(2)	0.000	0.000	0.004
(4.a)	1	125.920	1323	(3)	0.000	0.000	0.004
(4.b)	1	4.020	10	(4)	0.000	0.003	0.036

(1) construction of network  $(\mathcal{N}', \mathcal{A}')$ , (2) finding the SDPSP concerning  $c_1$ , (3) finding the SDPSP concerning  $c_2$ , (4) finding the non-dominated solutions, (4.a) generated deviation paths, (4.b) non-dominated solutions.

(a) Number of paths generated

(b) Running times (in seconds)

Table 4: Bicriteria non-dominated DPSP with MBA in random networks,  $n = 1000$ ,  $m = 4000$

pairs of simple paths, also for solving the bicriteria problem version MBA outperformed OBA as expected. So we decided to include in the paper the MBA results only. Tables 4 to 8 report the results obtained by MBA concerning two types of information, for each network size, the number of paths in the modified topology that had to be generated and the CPU times (in seconds) demanded by the algorithm. In both cases we refer to the four phases into which the algorithm is divided, namely: the construction of network  $(\mathcal{N}', \mathcal{A}')$ , the determination of the SDPSP concerning  $c_1$ ; the determination of the SDPSP concerning  $c_2$ ; and finally the ranking of DPSP concerning  $c_2$ , until all the non-dominated solution pairs have been found. The tables on the left (a) present the number of paths generated in the several phases of the algorithm, and the tables on the right (b) present the running times observed for each of the four phases.

	Min.	Average	Max.		Min.	Average	Max.
(2)	2	9.960	41	(1)	0.000	0.001	0.004
(3)	2	9.000	110	(2)	0.000	0.001	0.008
(4.a)	1	92.040	603	(3)	0.000	0.004	0.032
(4.b)	1	4.540	19	(4)	0.000	0.005	0.036

(1) construction of network  $(\mathcal{N}', \mathcal{A}')$ , (2) finding the SDPSP concerning  $c_1$ , (3) finding the SDPSP concerning  $c_2$ , (4) finding the non-dominated solutions, (4.a) generated deviation paths, (4.b) non-dominated solutions.

(a) Number of paths generated

(b) Running times (in seconds)

Table 5: Bicriteria non-dominated DPSP with MBA in random networks,  $n = 2000$ ,  $m = 8000$

	Min.	Average	Max.		Min.	Average	Max.
(2)	2	19.280	272	(1)	0.000	0.002	0.020
(3)	2	14.580	133	(2)	0.000	0.002	0.020
(4.a)	1	155.240	1055	(3)	0.000	0.001	0.008
(4.b)	1	5.540	14	(4)	0.000	0.005	0.032

(1) construction of network  $(\mathcal{N}', \mathcal{A}')$ , (2) finding the SDPSP concerning  $c_1$ , (3) finding the SDPSP concerning  $c_2$ , (4) finding the non-dominated solutions, (4.a) generated deviation paths, (4.b) non-dominated solutions.

(a) Number of paths generated

(b) Running times (in seconds)

Table 6: Bicriteria non-dominated DPSP with MBA in random networks,  $n = 3000$ ,  $m = 12000$

Even though the instances of this second set of tests were bigger than the instances of the first one, the computation of the SPDSP was still very fast, but in average up to 27 paths had to be listed before the best solution has been found and this took 0.003 seconds of CPU time. Also the number

	Min.	Average	Max.		Min.	Average	Max.
(2)	2	14.880	124	(1)	0.000	0.001	0.004
(3)	2	10.220	79	(2)	0.000	0.001	0.012
(4.a)	3	130.920	826	(3)	0.000	0.001	0.004
(4.b)	1	5.313	12	(4)	0.000	0.008	0.040

(1) construction of network  $(\mathcal{N}', \mathcal{A}')$ , (2) finding the SDPSP concerning  $c_1$ , (3) finding the SDPSP concerning  $c_2$ , (4) finding the non-dominated solutions, (4.a) generated deviation paths, (4.b) non-dominated solutions.

(a) Number of paths generated

(b) Running times (in seconds)

Table 7: Bicriteria non-dominated DPSP with MBA in random networks,  $n = 4000$ ,  $m = 16000$

	Min.	Average	Max.		Min.	Average	Max.
(2)	2	21.800	108	(1)	0.000	0.001	0.004
(3)	2	26.080	512	(2)	0.000	0.003	0.016
(4.a)	12	258.660	4611	(3)	0.000	0.002	0.060
(4.b)	1	6.120	15	(4)	0.000	0.015	0.072

(1) construction of network  $(\mathcal{N}', \mathcal{A}')$ , (2) finding the SDPSP concerning  $c_1$ , (3) finding the SDPSP concerning  $c_2$ , (4) finding the non-dominated solutions, (4.a) generated deviation paths, (4.b) non-dominated solutions.

(a) Number of paths generated

(b) Running times (in seconds)

Table 8: Bicriteria non-dominated DPSP with MBA in random networks,  $n = 5000$ ,  $m = 20000$

of path generated at phase (4) depended directly on the number of network nodes, the CPU times showed a similar behaviour, although they were more sensitive to the size of the problem. In average for the bigger instances ( $n = 5000$ ) the set of bicriteria DPSP demanded 258 path to be computed, 6 of them being non-dominated, in 0.015 seconds. These results seem suitable for both offline and real-time applications. In fact, as mentioned in the introduction this approach was used in the context of a traffic splitting problem in MPLS networks, aiming the minimisation of two objective functions: the number of arcs in the route from the origin to the destination and a measure of the load balancing cost of routes, subject to additional constraints related with the transmission delay and the available bandwidth in the arcs to be used. The method developed here was adapted for this traffic splitting problem and tested on networks with up to 100 nodes with satisfactory results [3].

## 6 Conclusions

In this paper we proposed a method to find non-dominated SDPSP when two additive objective functions are considered. This problem arose when dealing with the application of a multicriteria routing model for MPLS networks with traffic splitting, that demands DPSP to be used. The method presented is based on a procedure that allows DPSP to be ranked by non-decreasing order of cost, after a proper change in the topology of the given network, that is combined with a dominance test to filter only the non-dominated solutions. This method was tested in randomly generated networks of different sizes and that made it possible to compute all non-dominated solutions in 5000 nodes networks in an average time of about 0.015 seconds.

## References

- [1] Ramesh Bhandari. *Survivable Networks: Algorithms for Diverse Routing*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.
- [2] J. Clímaco and J. Craveirinha. Multiple criteria decision analysis - state of the art surveys. In J. Figueira, S. Greco, and M. Erghott, editors, *Int. Series in Operations Research and Management Science*, volume 78, chapter Multicriteria Analysis in Telecommunication Network Planning and Design - Problems and Issues, pages 899–951. Springer, 2005.
- [3] J. M. F. Craveirinha, Jo ao C. N. Clí maco, M. M. B. Pascoal, and L. M. R. A. Martins. Traffic splitting in MPLS networks - a hierarchical multicriteria approach, July 2007. VI International Conference on Decision Support for Telecommunications and Information Society - DSTIS'2007, Warsaw, Poland.
- [4] J. C. N. Clímaco and E. Q. V. Martins. A bicriterion shortest path algorithm. *European Journal of Operational Research*, 11:399–404, 1982.
- [5] E. Q. V. Martins, M. M. B. Pascoal, and J. L. E. Santos. Deviation algorithms for ranking shortest paths. *The International Journal of Foundations of Computer Science*, 10(3):247–263, 1999. (<http://www.mat.uc.pt/~marta/Publicacoes/deviation.ps.gz>).
- [6] J. W. Suurballe. Disjoint paths in a network. *Networks*, 4:125–145, 1974.
- [7] J. W. Suurballe and R. E. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14:325–336, 1984.