

FINDING NON-DOMINATED SHORTEST PAIRS OF DISJOINT SIMPLE PATHS

JOÃO C. N. CLÍMACO^(1,2), and MARTA M. B. PASCOAL^(2,3)

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

⁽²⁾ Instituto de Engenharia de Sistemas e Computadores – Coimbra, Rua Antero de Quental, 199, 3000-033 Coimbra, Portugal
E-mail: *jclimaco@inescc.pt*

⁽³⁾ Department of Mathematics, University of Coimbra, Apartado 3008, 3001-454 Coimbra, Portugal
E-mail: *marta@mat.uc.pt*

March 2007

Abstract: In this paper we introduce a method to compute non-dominated bicriteria shortest pairs, each including two disjoint simple paths. The method is an algorithm for ranking pairs of disjoint simple paths by non-decreasing order of cost, that is based on a ranking paths 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

Disjoint paths, that is paths that have no intermediate nodes in common, have a wide range of applications, for instance in telecommunication problems. The purpose is to find routes between source-destination pairs of nodes that don't share network resources and with minimum cost. The determination of $k \in \mathbb{N}$ disjoint simple paths, $k > 1$, has been considered by Suurballe, Suurballe and Tarjan, and Bhandari [6, 7, 1]. They formulated the problem as a minimum cost flow problem and proposed the application of a labeling algorithm, changing the given network.

This work was motivated by an application to a multicriteria routing model for MPLS (Multiprotocol Label Switching) Networks with traffic splitting in which pairs of disjoint simple paths have to be calculated. Two objective functions are involved in this model, namely the cost of using the two paths and the number of arcs in the paths [2]. The purpose of the paper is to develop a method to find the bicriteria shortest pairs of disjoint simple paths. We start by listing pairs of disjoint paths by non-decreasing order of a cost objective function, in order to deduce a way to generate all the non-dominated solutions when considering two costs.

The paper has four more sections. Section 2 introduces notation and the problem addressed. Section 3 proposes an algorithm for listing pairs of disjoint simple paths by order of cost, after a proper modification in the network topology. Section 4 focuses on a method for finding non-dominated pairs of disjoint simple paths 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.

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 pair of disjoint simple paths from s to t that minimise both objective functions.

3 Determination of disjoint pairs of simple paths

The determination of disjoint pairs of simple paths is discussed in this section. The first part concerns a modification of the given graph, aiming at transform pairs of simple paths between a pair of nodes into simple paths of the modified network. The second part gives a method for listing disjoint pairs of simple paths by non-decreasing order of a cost function, providing an algorithm for finding them.

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.

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' don't share nodes and they correspond to a pair of disjoint simple paths in $(\mathcal{N}, \mathcal{A})$. The following definitions intend to simplify the implementation of a method to find the shortest pair of disjoint simple paths.

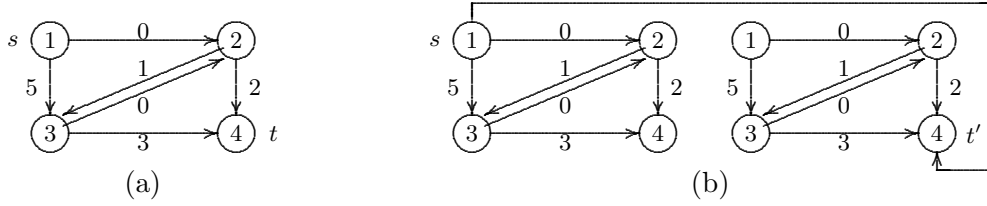


Figure 1: Networks (a) $(\mathcal{N}, \mathcal{A})$ and (b) $(\mathcal{N}', \mathcal{A}')$

Definition 3 A path p is simple iff $x \notin \text{sub}_p(s, x)$, for any node $x \in p$ and any node $y \in p$ previous to x .

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

Using a ranking algorithm we can list the simple paths from s to t' , and check whether each path in the new network corresponds to a disjoint pair of paths. This allows to filter only the disjoint pairs of paths, ordering them by cost. In such a process equivalent pairs of paths, as (p, q') and (q, p') , can be generated and one of them should be discarded.

Two remarks should be made regarding the memory space demanded to store the modified network $(\mathcal{N}', \mathcal{A}')$. The number of nodes in \mathcal{N}' is $2n$. The number of arcs in \mathcal{A}' is $2m + 1$ but the duplication of arcs can be avoided if the correspondent arcs in \mathcal{A} are analysed whenever an arc in \mathcal{A}' is considered, therefore only $m + 1$ arcs need to be stored.

3.2 Ranking disjoint pairs of simple paths by cost

The ranking of disjoint pairs of paths 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 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) doesn't 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 . 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 shouldn't 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 pair of disjoint paths 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's no pair of disjoint simple paths */
 $\bar{c}_{ij} \leftarrow \pi_i - \pi_j + c_{ij}, \forall (i, j) \in \mathcal{A}'$
Represent \mathcal{A}' in the sorted forward star form concerning \bar{c}
 $d_p \leftarrow s; X \leftarrow \{p\}; feasible \leftarrow \text{False}$
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 (sub $_p(s, v_i)$ is simple) and *disjoint*(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 sub $_p(s, v_i)$) or not *disjoint*(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 $a_l; q \leftarrow$ 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's no pair of disjoint simple paths */

4 Method for finding non-dominated solutions

In the bicriteria problem of finding non-dominated disjoint pairs of simple paths we intend to find disjoint pairs of simple paths 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 disjoint pairs of simple paths such that there is no other which improves one objective function without worsening the other.

Given (p_i, q_i) , $i = 1, 2$, two disjoint pairs of simple paths 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's no disjoint pair of simple paths 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 pairs of disjoint simple paths. 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 costs (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 costs (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 shortest disjoint pair of paths 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 $c_1(p) = c_1^*$, while the last one is such that $c_2(p) = \hat{c}_2$. The scheme bellow summarises the method to find non-dominated solutions, using the procedure described in the previous section to rank disjoint pairs of paths by cost.

- Find the shortest simple path from s to t' in $(\mathcal{N}', \mathcal{A}')$ concerning c_2 corresponding to a disjoint pair of simple paths 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 disjoint pair of simple paths 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 disjoint pair of simple paths 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

 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

/* Dominance test */

```

         $\mathcal{P}_X \leftarrow \{p_k\}; m_2 \leftarrow c_2(p_k)$ 
    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 method proposed to compute the set of non-dominated pairs of disjoint simple paths was implemented in C language and tested on randomly generated networks. Some computational results of these experiments were run on a Core 2 at 1.66 GHz, with 1 Gb of RAM, and are presented in Tables 1–4.

	Min.	Average	Max.		Min.	Average	Max.
(2)	8	50.721	301	(1)	0.000	0.000	0.000
(3)	8	35.837	150	(2)	0.000	0.000	0.000
(4.a)	0	1776.070	9932	(3)	0.000	0.000	0.004
(4.b)	1	3.930	9	(4)	0.000	0.167	6.432

(1) construction of network $(\mathcal{N}', \mathcal{A}')$, (2) finding the shortest pair of disjoint simple paths concerning c_1 ,
(3) finding the shortest pair of disjoint simple paths 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 1: Random network with $n = 50$ and $m = 200$

The reported results for each tested network size concern two types of information, the number of paths in the modified topology that have 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 shortest pair of disjoint simple paths concerning c_1 ; the determination of shortest pair of disjoint simple paths concerning c_2 ; and finally the ranking of shortest pairs of disjoint simple paths concerning c_2 , until all the non-dominated solution pairs have been found.

Random networks with 50, 100, 500 and 1000 nodes and $m = 4n$ arcs were generated. The program ran in 50 instances corresponding to 50 initial-terminal pairs of nodes for each considered size. In each table we present the minimum, average and maximum results for 50 instances generated for each of the network sizes. The left table (a) presents the number of paths generated in the several phases of the algorithm, and the right table (b) presents the running times observed for each of the four phases.

From the obtained results in the tables one can notice the increase in the number of paths the algorithm had to store with the size of the network. An upperbound on this number of paths was imposed on the program that was developed (30000880 paths in the presented experiments), and for some of the problems this limit wasn't enough to guarantee all the non-dominated solutions have been found, while in other cases this procedure wasn't able to calculate all the non-dominated solutions. However, the two considered objective functions are additive, and then each of them can be used for leading the ranking of disjoint pairs of simple paths while filtering the non-dominated solutions. This made it possible to apply the proposed method twice, after switching the roles of the two cost

	Min.	Average	Max.		Min.	Average	Max.
(2)	9	96.354	493	(1)	0.000	0.000	0.000
(3)	9	83.208	921	(2)	0.000	0.000	0.000
(4.a)	0	477063.906	12912779	(3)	0.000	0.000	0.000
(4.b)	1	5.625	12	(4)	0.000	0.682	16.134

(1) construction of network $(\mathcal{N}', \mathcal{A}')$, (2) finding the shortest pair of disjoint simple paths concerning c_1 , (3) finding the shortest pair of disjoint simple paths 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 2: Random network with $n = 100$ and $m = 400$

	Min.	Average	Max.		Min.	Average	Max.
(2)	10	432.550	13455	(1)	0.000	0.000	0.000
(3)	12	413.267	13455	(2)	0.000	0.001	0.020
(4.a)	1	7079275.000	30000880	(3)	0.000	0.001	0.020
(4.b)	2	7.432	14	(4)	0.000	20.015	96.074

(1) construction of network $(\mathcal{N}', \mathcal{A}')$, (2) finding the shortest pair of disjoint simple paths concerning c_1 , (3) finding the shortest pair of disjoint simple paths 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 3: Random network with $n = 500$ and $m = 2000$

functions, whenever the code has been interrupted because the storage limit was exceeded. After repeating the algorithm application for the necessary cases it was possible to find the whole set of non-dominated solutions for 44, 48, 44 and 42 problems in the 50, 100, 500 and 1000 node networks, respectively¹. Finally, it should be noticed that for some of the incomplete problems one cannot assure that a feasible solution indeed exists, that is, that there is at least one pair of disjoint simple paths from the origin to the destination.

	Min.	Average	Max.		Min.	Average	Max.
(2)	15	78.633	794	(1)	0.000	0.000	0.000
(3)	14	84.878	554	(2)	0.000	0.000	0.004
(4.a)	0	2866959.000	30000812	(3)	0.000	0.000	0.004
(4.b)	2	7.077	14	(4)	0.000	21.690	204.898

(1) construction of network $(\mathcal{N}', \mathcal{A}')$, (2) finding the shortest pair of disjoint simple paths concerning c_1 , (3) finding the shortest pair of disjoint simple paths 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: Random network with $n = 1000$ and $m = 4000$

It must be emphasised that the proposed network topology modification seems to make the bicriteria problem harder to solve, as similar techniques have been applied, with more effective results, to determine bicriteria simple paths – see [3]. In fact, when the original graph is duplicated not only the size of the problem and the number of solutions is doubled, but also the arc (t, s') linking the two subnetworks (which is included in any path from s to t') acts as a bottleneck for its deviation paths, and eventually increases the number of paths that have to be generated until a new feasible solution

¹As the code ran for 50 instances, 88%, 96%, 88% and 84% problems have been fully solved for networks with 50, 100, 500 and 1000 nodes, respectively.

is found.

The values in the tables report only those instances where the entire set of non-dominated solutions was determined, including the cases that demanded the leading cost function to be switched. These were the cases that achieved the maximum results that appear on the tables and, even though they influenced the average, the worst and the average values were still very different. In the tests performed, the computation of the shortest pair of disjoint simple paths was very fast, regardless of the objective function considering for performing the ranking, and the number of deviations generated in phase (4) (computing the non-dominated solutions set), as well as the correspondent running times, increased with the size of the network. The two worst cases occurred for the networks with 500 and with 1000 nodes, and it took about 20 seconds to find those sets, though it should be remarked that these values include the instances for which the upperbound on the number of stored paths was exceeded.

6 Conclusions

In this note we proposed a method to find non-dominated shortest pairs of disjoint simple paths 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 disjoint pairs of simple paths to be used. The method presented is based on a procedure that allows disjoint pairs of simple paths 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 1000 nodes networks in an average time of about 20 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. C. N. Clímaco, J. M. F. Craveirinha, and M. M. B. Pascoal. A bicriterion approach for routing problems in multimedia networks. *Networks*, 11:399–404, 2003.
- [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.