Computers & Operations Research ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/cor

Finding non-dominated bicriteria shortest pairs of disjoint simple paths

João C.N. Clímaco[a,b], Marta M.B. Pascoal[b,c,*]

[a]Faculty of Economics, University of Coimbra, Avenida Dias da Silva, 165, 3004-512 Coimbra, Portugal
[b]Instituto de Engenharia de Sistemas e Computadores – Coimbra, Rua Antero de Quental, 199, 3000-033 Coimbra, Portugal
[c]Department of Mathematics, University of Coimbra, Apartado 3008, 3001-454 Coimbra, Portugal

ARTICLE INFO

Keywords:
Disjoint paths
Shortest simple paths
Bicriteria
Ranking algorithms

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 path ranking 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.

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, minimising 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 multiple routes, 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 specific 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 [1–3]. These approaches consist of formulating the problem as a minimum cost flow problem and propose the application of a labelling 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 paths by order of a cost function. This work was motivated by an application to a multicriteria routing model for Multiprotocol Label Switching MPLS networks with traffic splitting [4] in which pairs of disjoint simple paths (PDSP) are used to spread the traffic. This model involves two objective functions to be minimised, namely the cost of using the two paths, given by the summation of their arc costs, which measures the load balancing cost, and the number of arcs in the paths [4,5]. To deal with this traffic splitting problem a method for listing shortest pairs of disjoint simple paths (SPDSP) was developed, and as the aim was to minimise two objective functions we computed only the non-dominated pairs. Here we consider two additive cost functions. The purpose of the paper is to introduce a method to determine PDSP, which is also able of listing these paths by non-decreasing order of cost. Finally this ranking method is combined with a dominance test thus leading to a process to generate all the non-dominated solutions when two costs are considered.

The paper has four more sections. The following introduces some notation and the adequate problem. Section 3 proposes an algorithm for listing PDSP by order of cost after an adequate 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 $p = \langle i = v_1, v_2, \ldots, j = v_{\ell(p)} \rangle$, where $(v_k, v_{k+1}) \in \mathcal{A}$, for any $k \in \{1, \ldots, \ell(p) - 1\}$, where $l(p)$ is the number of nodes of path p. 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.

* Corresponding author at: Department of Mathematics, University of Coimbra, Apartado 3008, 3001-454 Coimbra, Portugal. Tel.: +351 239791150; fax: +351 239832568.
E-mail addresses: jclimaco@inescc.pt (J.C.N. Clímaco), marta@mat.uc.pt (M.M.B. Pascoal).

0305-0548/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2009.01.002

Please cite this article as: Clímaco JCN, Pascoal MMB. Finding non-dominated bicriteria shortest pairs of disjoint simple paths. Computers and Operations Research (2009), doi: 10.1016/j.cor.2009.01.002

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

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

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

Given $p\in\mathscr{P}_{xi}$ and $q\in\mathscr{P}_{iy}$ 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. Two costs are associated with each arc (x,y) of the network: $c^i_{xy}\in\mathrm{IR}$, $i=1,2$. Given any path p two objective functions are defined:

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

Our first aim is the determination of a PDSP from s to t that minimises both objective functions.

3. Determination of PDSP

The determination of PDSP is discussed in this section. The first part concerns a modification of the given graph, aiming at transforming pairs of simple paths between s and t into simple paths of the modified network. The second part gives a method for listing PDSP by non-decreasing order of a cost function. This algorithm ranks paths that correspond to pairs of paths in the original network, reducing as much as possible the calculation of pairs with nodes in common, and, finally, it filters the disjoint pairs.

3.1. Network topology modification

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

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

In the new network the initial node is still s and the new terminal node is t'. The costs of the original arcs are maintained, while for the new ones we have $c_{i'j'}=c_{ij}$, if $(i,j)\in\mathscr{A}$, and $c_{ts'}=0$. Fig. 1 shows an example of a network and its duplication.

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

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

where $q\in\mathscr{P}_{st}$ and $q'\in\mathscr{P}'_{s't'}$. If q and q' are simple and do not share corresponding nodes in $\mathscr{N}$ and $\mathscr{N}'$, then they are associated with a PDSP in $(\mathscr{N},\mathscr{A})$. The following definitions intend to simplify the implementation of a method to find the SPDSP.

Definition 3. A path p in $\mathscr{P}$ is simple iff for any node $x\in p$, if y is a node of p previous to x then $y\notin \mathrm{sub}_p(x,t)$.

Definition 4. Two simple paths $q_1,q_2\in\mathscr{P}$ 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 PDSP. This allows only the PDSP to be filtered, ordering them by cost.

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

3.2. Ranking disjoint pairs of simple paths by cost

The ranking of PDSP by non-decreasing order of cost can be made by using an adaptation of the algorithm by Martins, Pascoal and Santos [6] for ranking simple paths. This algorithm, now briefly reviewed, allows to easy the incorporation of 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, $k=1,\ldots,K$. The set X is initialised with the shortest path from s to t in $(\mathscr{N},\mathscr{A})$, p_1, and if $p_1,\ldots,p_{k-1}$ have been determined then p_k is the next shortest candidate in X. When p_k is selected and removed from X its nodes are analysed, in order to generate new candidates with a low cost. The shortest deviation of p_k at node v_i is given by the shortest path from v_i to t, after the arc of p_k that starts at v_i being deleted. This best path has the form:

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

where (v_i,j) does not belong to any of the candidates computed so far, $\mathscr{T}_t$ denotes the tree of shortest paths from any node to t, and $\mathscr{T}_t(j)$ denotes the path from j to t in $\mathscr{T}_t$. In the following π_j stands for $c(\mathscr{T}_t(j))$. 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 $\mathscr{N}$ and to $\mathscr{N}'$ should be distinct, except s,t,s',t'. This procedure may generate paths with loops or with common nodes in $\mathscr{N}$ and $\mathscr{N}'$ whenever $\mathrm{sub}_{p_k}(s,v_i)$ and $\mathscr{T}_{t'}(j)$ 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 [6]. Algorithm 1 refers to function *Disjoint*, that checks if a path p in $(\mathscr{N}',\mathscr{A}')$ corresponds to a PDSP in $(\mathscr{N},\mathscr{A})$. In other words this function checks whether the path $p\cap\mathscr{N}$ and the

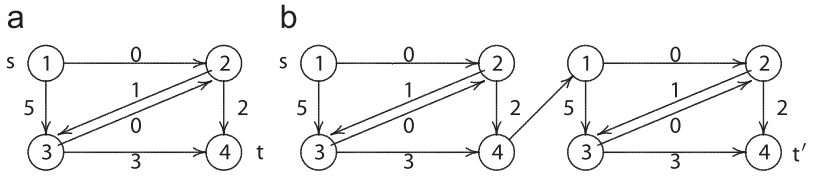

Fig. 1. Networks (a) $(\mathscr{N},\mathscr{A})$ and (b) $(\mathscr{N}',\mathscr{A}')$.

Please cite this article as: Clímaco JCN, Pascoal MMB. Finding non-dominated bicriteria shortest pairs of disjoint simple paths. Computers and Operations Research (2009), doi: 10.1016/j.cor.2009.01.002

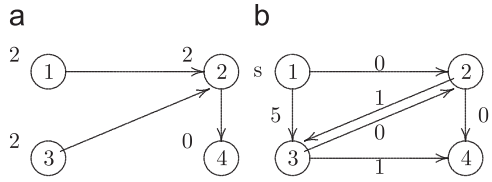

Fig. 2. (a) $\mathscr{T}_t$ in $(\mathscr{N},\mathscr{A})$ and (b) $(\mathscr{N},\mathscr{A})$ with reduced costs.

path that corresponds to $p \cap \mathscr{N}'$ in the original network share other nodes besides the initial and the terminal.

Algorithm 1. Determination of a shortest PDSP in terms of a cost function

$\mathscr{T}_{t'} \leftarrow$ tree of the shortest paths from $i \in \mathscr{N}'$ to t' in terms c
$p \leftarrow \mathscr{T}_{t'}(s)$
`If` (p is not defined) `Then Stop`
$\bar{c}_{ij} \leftarrow \pi_j - \pi_i + c_{ij}, \forall (i,j) \in \mathscr{A}'$
Represent $\mathscr{A}'$ in the sorted forward star form in terms of $\bar{c}$
$d_p \leftarrow s$; $X \leftarrow \{p\}$; *feasible* $\leftarrow$ `False`
`While` (($X \neq \emptyset$) `and` (`not` *feasible*)) `Do`
  $p \leftarrow$ path in X such that $\bar{c}(p)$ is minimum
  $X \leftarrow X - \{p\}$
  $i \leftarrow$ 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; $q \leftarrow \text{sub}_p(s, v_i) \diamond a_l \diamond \mathscr{T}_t(v_j)$; $d_q \leftarrow v_i$; $X \leftarrow X \cup \{q\}$
      `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`

Note that the first steps of MPS algorithm compute $\mathscr{T}_t$ and replace arc costs by reduced costs. The shortest tree from any node to the terminal and the new costs are depicted in Figs. 2 and 3 when considering the original network $(\mathscr{N},\mathscr{A})$ and the duplicated network $(\mathscr{N}',\mathscr{A}')$, respectively, both given in Fig. 1.

As shown in the pictures, it is also easy to prove

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

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

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 Fig. 1 Algorithm 1 first determines the shortest path from 1 to 4′, $p = \langle 1,2,4,1',2',4' \rangle$, which corresponds to the pair (q,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)

Hence, it is easy to see that 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 $c(\text{sub}_p(s,t)) \leqslant c(\text{sub}_p(s',t'))$ (or instead $c(\text{sub}_p(s,t)) \geqslant c(\text{sub}_p(s',t'))$, 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`
EquivalentPair ($\text{sub}_p(s, v_i) \diamond a_{l+1} \diamond \mathscr{T}_{t'}$(head node of a_{l+1})) `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`

where *EquivalentPair* stands for a boolean function that, given a path p from s to t', returns the value `True` iff

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

By applying this test equivalent pairs of paths, both with the same cost, can still be obtained. Extending it by replacing the strict inequality by $\geqslant$, in order to avoid such cases, may prevent other pairs of paths from being computed. Instead, those pairs can be identified if compared with the already computed ones, either when they are generated or else when they are selected in X.

Please cite this article as: Clímaco JCN, Pascoal MMB. Finding non-dominated bicriteria shortest pairs of disjoint simple paths. Computers and Operations Research (2009), doi: 10.1016/j.cor.2009.01.002

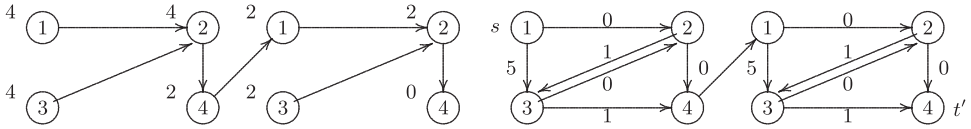

Fig. 3. (a) $\mathscr{T}_{t'}$ in $(\mathscr{N}', \mathscr{A}')$ and (b) $(\mathscr{N}', \mathscr{A}')$ with reduced costs.

Remark 3. It should also be noted that to allow ranking PDSP 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 $(\mathscr{N}, \mathscr{A})$ is replicated k times, then Algorithm 1 can be applied to determine the shortest set of k disjoint simple paths, $k \in$ IN.

In terms of the theoretical number of performed operations, the determination of $\mathscr{T}_{t'}$ can be made in $\mathscr{O}(m' + n' \log n')$, where n' and m' are the number of nodes and the number of arcs of $(\mathscr{N}', \mathscr{A}')$, if Dijkstra's algorithm [7] is used together with a Fibonacci heap [8]. Replacing arc costs by reduced costs and then sorting $\mathscr{A}$, in order to represent $(\mathscr{N}, \mathscr{A})$ in the sorted forward star form, takes $\mathscr{O}(m' \log n')$. Each `While` loop demands choosing the best path in X, scanning at most n nodes, selecting the next arc to generate a new candidate and inserting it in X, which can be made in $\mathscr{O}(m' + \log |X|)$. If K_1 paths are computed and the `While` loop is executed K_2 times, then the total number of operations executed by Algorithm 1 will be $\mathscr{O}(m' \log n' + K_2(m' + \log K_1))$. In general K_1, K_2 cannot be known in advance once the paths to be generated may not satisfy to the imposed constraints.

Table 1
CPU times for ORA ranking $K = 5000$ PDSP in random networks, $n = 1000$.

	Min.	Average	Max.
(a) $m = 5000$			
(1)	0.000	0.000	0.004
(2)	0.000	0.001	0.004
(3)	0.488	0.662	1.204
(b) $m = 10\,000$			
(1)	0.000	0.000	0.004
(2)	0.000	0.004	0.160
(3)	0.524	1.282	18.350
(c) $m = 20\,000$			
(1)	0.000	0.000	0.004
(2)	0.000	0.004	0.096
(3)	0.500	0.986	3.998

(1) Constructing $(\mathscr{N}', \mathscr{A}')$, (2) finding the best PDSP, (3) ranking 5000 solutions.

4. Method for finding non-dominated solutions

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

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

In 1982, Clímaco and Martins [9] introduced a method to solve the bicriteria shortest path problem. This method is adapted here to determine the set of non-dominated PDSP. The idea involved in their method is to list 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 $(\mathscr{N}', \mathscr{A}')$ is found the associated objective function values, $(c_1(p), c_2(p))$, have to be compared with the previously obtained ones. 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. 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 $c_2(p) \geqslant m_2$ 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. 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 $(\mathscr{N}', \mathscr{A}')$ that corresponds to a SPDSP in $(\mathscr{N}, \mathscr{A})$ 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 $c_1(p) \leqslant \hat{c}_1$ and $c_2(p) \geqslant c_2^*$. The scheme below summarises the method to find non-dominated solutions, using the procedure described in the previous section to rank PDSP by cost.

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

Please cite this article as: Clímaco JCN, Pascoal MMB. Finding non-dominated bicriteria shortest pairs of disjoint simple paths. Computers and Operations Research (2009), doi: 10.1016/j.cor.2009.01.002

Algorithm 2. Determination of non-dominated shortest PDSP

Construct the modified network $(\mathcal{N}', \mathcal{A}')$
$p^*_{c_2} \leftarrow$ shortest pair of disjoint simple paths in terms of c_2; $\hat{c}_1 \leftarrow c_1(p^*_{c_2})$
$p^*_{c_1} \leftarrow$ shortest pair of disjoint simple paths in terms of 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`

In [10] Hansen proved the bicriteria shortest path problem to be intractable. The number of operations that Algorithm 2 performs in order to obtain all non-dominated solutions depends on the ranking method—see Algorithm 1—as well as on the number of paths that has to be listed.

5. Computational tests

The approaches proposed in the previous sections to deal with PDSP 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 PDSP, the original method, ORA, and the modification that reduces the number of repeated pairs of paths, MRA, described in Remark 2. Equivalent PDSP with the same cost are filtered when they are selected in the set of candidates, X. Afterwards these codes were used to implement two versions of Algorithm 2, for computing the non-dominated PDSP, OBA and MBA, when considering the original and the modified methods, respectively, as above. In these implementations the whole set $\mathcal{A}'$ was maintained. In this section some computational results of these experiments on randomly generated networks are presented and discussed.

The first set of tests aimed to evaluate the applicability and the efficiency of the two versions of Algorithm 1 when extended to rank a given number of PDSP. The program ran on networks of 1000 and 2000 nodes and average degree $d = 5, 10, 20$ ($d = m/n$). For each network size 50 instances, corresponding to 50 initial–terminal pairs of nodes, were considered. The minimum, average and maximum CPU times demanded by ORA for ranking the 5000 shortest solutions in those 50 instances are presented in Tables 1 and 2.

For these instances both the conversion of the original network and the determination of the shortest PDSP were extremely fast. On the other hand, for 2000 node instances ORA was able to produce the 5000 best solutions in about 1–2.5 s. The CPU times seem to be more sensitive to the increase in the number of network nodes rather than to the network density.

Version MRA was also tested under the same conditions of the first set. Table 3 presents a comparison between the average processing times obtained with ORA and MRA for ranking 5000 solutions on this test bed. These results show the latter version, MRA, decreased the time for ranking 5000 PDSP in about 30%.

The second set of experiments tested the implementations of Algorithm 2, OBA and MBA, for finding the non-dominated PDSP. This set is composed of random networks with 1000, 2000, 3000, 4000 and 5000 nodes and $m = 4n$ arcs, and again 50 instances were considered for each size. As for ranking PDSP, as expected MBA outperformed OBA for solving the bicriteria problem, so we decided to include in the paper only the MBA results. Tables 4–8 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 SPDSP concerning c_1; the determination of the SPDSP concerning c_2; and finally the ranking of PDSP 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, while the tables on the right (b) present the running times observed for each of the four phases.

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, in average up to 27 paths had to be listed before the best solution has been found and this took 0.003 s of CPU time. Also the number 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 PDSP demanded 258 paths to be computed, 6 of them being non-dominated, in 0.015 s. These results are 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 at 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 to the transmission delay and the available bandwidth in the arcs to be used. The

Please cite this article as: Clímaco JCN, Pascoal MMB. Finding non-dominated bicriteria shortest pairs of disjoint simple paths. Computers and Operations Research (2009), doi: 10.1016/j.cor.2009.01.002

Table 2
CPU times for ORA ranking $K = 5000$ PDSP in random networks, $n = 2000$.

	Min.	Average	Max.
(a) $m = 10\,000$			
(1)	0.000	0.001	0.004
(2)	0.000	0.003	0.020
(3)	1.008	1.856	15.253
(b) $m = 20\,000$			
(1)	0.000	0.000	0.004
(2)	0.000	0.005	0.120
(3)	1.008	1.618	6.151
(c) $m = 40\,000$			
(1)	0.000	0.001	0.004
(2)	0.000	0.005	0.064
(3)	1.012	2.511	15.361

(1) Constructing $(\mathcal{N}', \mathcal{A}')$, (2) finding the best PDSP, (3) ranking 5000 solutions.

Table 3
Average CPU times improvement ($1 - \text{MRA}/\text{ORA}\%$) for ranking $K = 5000$ PDSP in random networks.

	$d = 5$	$d = 10$	$d = 20$
$n = 1000$	32.29	28.62	27.59
$n = 2000$	32.33	29.13	23.38

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

	Min.	Average	Max.
(a) Number of paths generated			
(2)	2	11.980	162
(3)	2	14.600	156
(4.a)	1	125.920	1323
(4.b)	1	4.020	10
(b) Running times (in seconds)			
(1)	0.000	0.001	0.004
(2)	0.000	0.000	0.004
(3)	0.000	0.000	0.004
(4)	0.000	0.003	0.036

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

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

	Min.	Average	Max.
(a) Number of paths generated			
(2)	2	9.960	41
(3)	2	9.000	110
(4.a)	1	92.040	603
(4.b)	1	4.540	19
(b) Running times (in seconds)			
(1)	0.000	0.001	0.004
(2)	0.000	0.001	0.008
(3)	0.000	0.004	0.032
(4)	0.000	0.005	0.036

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

method developed here was adapted for this traffic splitting problem and tested on networks with up to 100 nodes with satisfactory results [4].

Table 6
Bicriteria non-dominated PDSP with MBA in random networks, $n = 3000$, $m = 12\,000$.

	Min.	Average	Max.
(a) Number of paths generated			
(2)	2	19.280	272
(3)	2	14.580	133
(4.a)	1	155.240	1055
(4.b)	1	5.540	14
(b) Running times (in seconds)			
(1)	0.000	0.002	0.020
(2)	0.000	0.002	0.020
(3)	0.000	0.001	0.008
(4)	0.000	0.005	0.032

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

Table 7
Bicriteria non-dominated PDSP with MBA in random networks, $n = 4000$, $m = 16\,000$.

	Min.	Average	Max.
(a) Number of paths generated			
(2)	2	14.880	124
(3)	2	10.220	79
(4.a)	3	130.920	826
(4.b)	1	5.313	12
(b) Running times (in seconds)			
(1)	0.000	0.001	0.004
(2)	0.000	0.001	0.012
(3)	0.000	0.001	0.004
(4)	0.000	0.008	0.040

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

Table 8
Bicriteria non-dominated PDSP with MBA in random networks, $n = 5000$, $m = 20\,000$.

	Min.	Average	Max.
(a) Number of paths generated			
(2)	2	21.800	108
(3)	2	26.080	512
(4.a)	12	258.660	4611
(4.b)	1	6.120	15
(b) Running times (in seconds)			
(1)	0.000	0.001	0.004
(2)	0.000	0.003	0.016
(3)	0.000	0.002	0.060
(4)	0.000	0.015	0.072

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

6. Conclusions

In this paper we proposed a method for finding non-dominated PDSP when two additive objective functions are considered. This problem arose when dealing with the application of a routing model for MPLS networks with traffic splitting that involves two objective functions and demands PDSP to be used. The method presented is based on a procedure that allows PDSP to be ranked by non-decreasing order of cost after a proper change in the topology of the given network, and 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 node networks in an average time of about 0.015 s.

Please cite this article as: Clímaco JCN, Pascoal MMB. Finding non-dominated bicriteria shortest pairs of disjoint simple paths. Computers and Operations Research (2009), doi: 10.1016/j.cor.2009.01.002

In the application that motivated this study are involved two criteria and the determination of pairs of disjoint paths. Possible aspects for future research on this topic include the extension of the presented ideas combined with the method proposed in [11], in a context involving more than two criteria, and the evaluation of this method when applied to the determination of sets of more than two disjoint simple paths.

References

[1] Bhandari R. Survivable networks: algorithms for diverse routing. Norwell, MA, USA: Kluwer Academic Publishers; 1998.
[2] Suurballe J. Disjoint paths in a network. Networks 1974;4:125–45.
[3] Suurballe J, Tarjan R. A quick method for finding shortest pairs of disjoint paths. Networks 1984;14:325–36.
[4] Craveirinha J, Clímaco J, Pascoal M, Martins L. Traffic splitting in MPLS networks—a hierarchical multicriteria approach. In: VI international conference on decision support for telecommunications and information society—DSTIS'2007, Warsaw, Poland, July 2007.
[5] Clímaco J, Craveirinha J. Multiple criteria decision analysis—state of the art surveys. In: Figueira J, Greco S, Erghott M, editors. Multicriteria analysis in telecommunication network planning and design—problems and issues. International series in operations research and management science, vol. 78. Berlin: Springer; 2005. p. 899–951.
[6] Martins E, Pascoal M, Santos J. Deviation algorithms for ranking shortest paths. The International Journal of Foundations of Computer Science 1999;10(3):247–63.
[7] Dijkstra E. A note on two problems in connection with graphs. Numerical Mathematics 1959;1:269–71.
[8] Fredman M, Tarjan R. Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the Association for Computing Machinery 1987;34:596–615.
[9] Clímaco J, Martins E. A bicriterion shortest path algorithm. European Journal of Operational Research 1982;11:399–404.
[10] Hansen P. Bicriterion path problems. In: Fandel G, Gal T, editors. Multiple criteria decision making: theory and applications. Lectures notes in economics and mathematical systems, vol. 177. Berlin: Springer; 1980. p. 109–27.
[11] Clímaco J, Martins E. On the determination of the nondominated paths in a multiobjective network problem. Proceedings of the V sympösium über operations research, Köln, 1980, Methods in operations research, vol. 40. Königstein: Anton Hain; 1981. p. 255–8.

Please cite this article as: Clímaco JCN, Pascoal MMB. Finding non-dominated bicriteria shortest pairs of disjoint simple paths. Computers and Operations Research (2009), doi: 10.1016/j.cor.2009.01.002