ABSTRACT
This paper presents a new algorithm to compute the set of Pareto solutions in the multiobjective shortest path problem. The main idea consists of ranking lexicographic shortest paths using the Optimality Principle. Some computational results comparing this strategy with the labelling one are also reported.

Keywords: multiple objective programming, ranking algorithm, lexicographic order, nondominated path.

Corresponding author: José Luis Esteves dos Santos (zeluis@mat.uc.pt).
Departamento de Matemática, Universidade de Coimbra,
Apartado 3008, 3001-454 Coimbra, Portugal.
Fax number: 00 351 239832568
1. Introduction

The multiobjective shortest path problem (MSPP) is a natural extension of the classical shortest path problem when several parameters are assigned to the arcs of the underlying network.

The MSPP has been dealt with by many authors addressing either theoretical aspects, algorithmic approaches or applications. In particular, the bi-objective case has been extensively studied in the literature. For a review of the MSPP, the interested reader is referred to [8].

In the MSPP, one intends to determine a path that minimizes simultaneously all the criteria under consideration. Usually, there is a conflict among the different criteria and such an ideal solution does not exist. The resolution of the MSPP turns into finding nondominated paths (ND paths), that is, paths for which there is no other path with better values for all the criteria.

In the worst case, as proved by Hansen, [12], one may have to deal with an exponential number of ND paths. Therefore, the computation of the entire set of nondominated paths can be hard to accomplish.

Several strategies have been adopted for tackling the MSPP. As proposed by Martins and Santos, [17], those approaches can be classified into two classes. The first one groups the algorithms that select a specific ND path as is the case of a global optimization problem where an utility function is defined, [4]. Interactive procedures, [5], where the user is guiding the search into the criteria space, are also considered in this group. In this paper, we are interested in the second class of approaches for the MSPP where the full set of nondominated paths is obtained. For that purpose, two techniques have been considered in the literature: the well-known labelling algorithm, [3, 12, 22, 23] and the ranking paths procedures, [6, 7]. We must refer that Mote [19] presented an algorithm that computes the complete nondominated set after solving the linear programming relaxation for the MSPP.

In this paper, we present a new labelling technique based on a ranking path procedure where labels are scanned tracing the nodes of the next shortest deviation.
A new algorithm for the MSPP

path. The computational experience reported in section 6 of the paper, shows that the new algorithm is competitive relatively to the previous approaches. The larger the number of criteria, the more competitive the new algorithm is relatively to the labelling approaches.

Let us note now that, for the present study, we used an enhanced version of the algorithm presented by Skriver and Andersen, [22]. Actually, as can be seen in point 6.1 of this paper, our version shows to be much faster than the aforementioned one.

In the next section of the paper, we introduce the notation used through out the paper. The section 3 is dedicated to the description of the algorithm that is exemplified in the following section. The correctness of the algorithm is proved in section 5 and, as already mentioned, computational results are shown in section 6. Finally, conclusions are presented in section 7.

2. Definitions and notation

In this section, some definitions are given and only a crucial result is presented taking into account that detailed mathematical background can be found in Martins and Santos, [17].

A network is denoted by $\mathcal{G} = (\mathcal{N}, \mathcal{A}, c)$, where $\mathcal{N} = \{1, \ldots, n\}$ is the set of nodes (or vertices) and $\mathcal{A} = \mathcal{N} \times \mathcal{N}$ is the set of arcs. Each arc $a \in \mathcal{A}$, $a = (i, j)$, has a tail $(\text{tail}(a) = i)$ and a head $(\text{head}(a) = j)$ node. The set of arcs which tail node is $i$ will be denoted by $\mathcal{A}(i) = \{(x, y) \in \mathcal{A} : x = i\}$. Let $k$ be the number of criteria, then the vectorial function $c$ attributes a $k$ dimensional vector cost to each arc:

$$c : \mathcal{A} \rightarrow \mathbb{R}^k$$

$$(i, j) \rightarrow c(i, j) = c_{i,j} = (c^1_{i,j}, \ldots, c^k_{i,j}).$$

A path $p$, from the vertex $i$ to $j$, is an alternating sequence of nodes and arcs of the form $p = \langle v_0, a_1, v_1, \ldots, a_r, v_r \rangle$, where:

- $v_\ell \in \mathcal{N}$, $\forall \ell \in \{0, \ldots, r\}$;
- $v_0 = i$ and $v_r = j$;
- $a_\ell = (v_{\ell-1}, v_\ell) \in \mathcal{A}$, $\forall \ell \in \{1, \ldots, r\}$. 
A new algorithm for the MSPP

The set of all paths from \(i\) to \(j\) is denoted by \(P_{i,j}\) and \(P_G\) represents the set of all paths in the network, that is, \(P_G = \bigcup_{i,j \in \mathcal{N}} P_{i,j}\). A cycle is a path with non repeated vertices except the initial and terminal ones which are coincident; that is, \(v_0 = v_r\).

With no loss of generality, we consider that \(\mathcal{N}\) has an initial node \(s\) and a terminal node \(t\) such that:

- for any arc \(a \in A\), \(\text{tail}(a) \neq s\) and \(\text{head}(a) \neq t\);
- for any \(i \in \mathcal{N} - \{s, t\}\), \(P_{s,i} \neq \emptyset\) and \(P_{i,t} \neq \emptyset\).

In order to simplify the notation, \(P\) will be used instead of \(P_{s,t}\).

Multiple arcs (arcs with the same pair of head and tail nodes) are not allowed. As a consequence, \(p\) can be denoted only by the sequence of its nodes, \(\langle v_0, v_1, \ldots, v_r \rangle\).

We denote by \(\text{sub}_p(u, w)\) the subpath of \(p\) from \(u\) to \(w\) (\(u, w \in \mathcal{N} \cap p\)), that is, the subsequence \(\langle v_{\ell}, v_{\ell+1}, \ldots, v_{\ell+h} \rangle\), where \(v_{\ell} = u\) and \(v_{\ell+h} = w\).

The vectorial objective function \(f\) is defined by

\[
\begin{align*}
  f : P_G & \rightarrow \mathbb{R}^k \\
  p & \mapsto f(p) = (f_1(p), \ldots, f_k(p)),
\end{align*}
\]

where \(f_\ell(p) = \sum_{(i,j) \in p} c_{i,j}^\ell, \forall \ell \in \{1, \ldots, k\}\).

The concatenation operator, \(\diamond\), joins two paths \(p = \langle v_0, \ldots, v_{r_p} \rangle\) and \(q = \langle u_0, \ldots, u_{r_q} \rangle\) such that \(v_{r_p} = u_0\). Then, \(p \diamond q = \langle v_0, \ldots, v_{r_p} = u_0, \ldots, u_{r_q} \rangle\).

Now, that us recall that, in the MSPP, one looks for the set of nondominated paths from \(s\) to \(t\), mathematically described as follows:

**Definition 1**: Let \(p\) and \(q\) be two paths of \(P_{i,j}\). We say that \(p\) dominates \(q\) or \(q\) is dominated by \(p\) \((p <_D q)\) if and only if

\[
  f(p) \neq f(q) \quad \text{and} \quad f_\ell(p) \leq f_\ell(q), \forall \ell \in \{1, \ldots, k\}.
\]

**Definition 2**: Let \(p\) be a path in \(P_{i,j}, i, j \in \mathcal{N}\). If there is no path \(q \in P_{i,j}\) such that \(q <_D p\), then \(p\) is called nondominated, efficient or Pareto optimal path. The set of nondominated paths from \(i\) to \(j\) is denoted by \(\bar{D}_{i,j}\) and \(\bar{D}\) will be used for \(\bar{D}_{s,t}\).
Note that the dominance relation ($<_D$) is not a total order relation in $\mathbb{R}^k$ and, therefore, does not allow the full ranking of the paths in the network. Nevertheless, this may be achieved by considering the total order relation [9] established by the following:

**Definition 3**: Let $p$ and $q$ be two paths of $P_{i,j}$. Then, $p$ is lexicographically less than or equal to $q$ ($p \leq_L q$) if and only if $f(p) = f(q)$ or

$$\exists x \in \{1, \ldots, k\}: f_x(p) < f_x(q) \text{ and } f_y(p) = f_y(q), \forall y < x.$$ 

The above definitions straightforwardly lead to the next result that will be essential for proving the correctness of the ranking algorithm.

**Lemma 1**: Let $p, q \in P_{s,i}$ be two paths from $s$ to $i$ in $G$. If $p <_D q$ then $p \leq_L q$.

**Proof**: It is a trivial consequence from the definitions of $p <_D q$ and of $p \leq_L q$. \square

Note that other total order relations in $\mathbb{R}^k$ could be used for ranking the paths. For instance, one can consider the following:

- $p \leq_{\text{sum}} q \iff \sum_{x=1}^k f_x(p) \leq \sum_{x=1}^k f_x(q)$ (or other weighted sum);
- $p \leq_{\text{max}} q \iff \max\{f_x(p) : 1 \leq x \leq k\} \leq \max\{f_x(q) : 1 \leq x \leq k\}$

Experience reported in [21], show that those two order relations produce similar results to the lexicographic relation.

3. The label-deviation path algorithm (L&DP)

The new algorithm for the MSPP, presented in this paper, is based on the procedure for ranking paths developed by Martins et al., [15], for the shortest path problem with a single objective. That procedure consists of sequentially determining deviation paths in order to consider a non-decreasing sequence of shortest paths from $s$ to $t$. Next, we remind some basic aspects of the procedure and two results stated for the single objective case that will be used in the new label-deviation path algorithm.
Now, let us consider a path \( p = \langle v_0, v_1, \ldots, v_r \rangle \in \mathcal{P} \) and let \( v_i \) be a node of \( p \) \((0 \leq i < r)\). A deviation path from \( p \) at vertex \( v_i \) through the arc \((v_i, j) \in \mathcal{A}(v_i)\) is a path \( q \in \mathcal{P} \) which coincides with \( p \) from \( s = v_0 \) to \( v_i \), then follows arc \((v_i, j)\) and finally goes to \( v_r = t \) (see Figure 1 (a)). So, \( q = \text{sub}_p(s, v_i) \bowtie (v_i, j) \bowtie w \), where \( w \) is a path of \( \mathcal{P}_{j,t} \). Note that, \((v_i, j)\) is the first arc for which \( q \) diverges from \( p \). This arc is called the deviation arc of \( q \) relatively to \( p \).

For the single objective case, if the least cost path from \( j \) to \( t \) is denoted by \( T^*(j) \) then \( q^p_{v_i,j} = \langle v_0, \ldots, v_i \rangle \bowtie (v_i, j) \bowtie T^*(j) \) is the shortest deviation path from \( p \) through the arc \((v_i, j)\).

Hence, if one considers the vertex \( v_i \in p \), the shortest deviation path from \( p \) at the node \( v_i \) is the one corresponding to \( \min_{(v_i, j) \in \mathcal{A}(v_i)} f(q^p_{v_i,j}) \), (see Figure 1 (b)). Therefore, the shortest deviation path from \( p \) will be determined by \( \min_{v_i \in p} \left( \min_{(v_i, j) \in \mathcal{A}(v_i)} f(q^p_{v_i,j}) \right) \).

From above, one may depict a very simply procedure for generating shortest paths from \( s \) to \( t \), ordered by non-decreasing costs. The first path \((p^*_1)\) is the shortest path from \( s \) to \( t \) and the next path in the sequence \((p^*_2)\) will be the least cost path in \( D(p^*_1) = \{ q^{p^*_1}_{v_i,j} : v_i \in p^*_1, (v_i, j) \in \mathcal{A} \} \), the set of deviation paths from \( p^*_1 \). Then, \( p^*_3 \) will be obtained considering all the deviation paths from \( p^*_1 \) and \( p^*_2 \), excluding \( p^*_2 \), and the algorithm carry on, in this way and successively, until all the shortest paths are generated.

Since \( p^*_3 \) is itself a deviation path from \( p^*_1 \) through an arc \((v_i, j)\) with \( v_i \in p \), that we denote by \( \theta(p^*_3) \), we only need to consider the deviation paths from \( p^*_2 \) related to
A new algorithm for the MSPP

the vertices in \( p_2^* \), sequent to the tail of \( \theta(p_2^*) \), inclusive, since the remained deviation paths are obtained from \( p_3^* \). Therefore, \( p_3^* \) is obtained from \( D(p_1^*) \setminus \{p_2^*\} \) or \( D(p_2^*) = \{ q_{v_i,j}^p : v_i \in \text{sub}_p^2(\text{tail}(\theta(p_2^*)), t), (v_i, j) \in \mathcal{A} \} \). Again, \( \theta(p_3^*) \) will be the deviation arc through which \( p_3^* \) is generated and, consequently, defines \( D(p_3^*) \) the set of deviation paths from \( p_3^* \) that must be considered, together with \( (D(p_1^*) \cup D(p_2^*)) \setminus \{p_2^*, p_3^*\} \), for determining the next shortest path.

Next, we show how to reduce the computational effort for the calculation of \( \min_{(v_i, j) \in \mathcal{A}(v_i)} f(q_{v_i,j}) \) required for each vertex \( v_i \) in the successive \( D(p_\ell^*) \), \( \ell \geq 1 \).

Let us recall that \( T^*(i) \) is the shortest path from \( i \) to \( t \). Those paths can be chosen in such a way that \( T^* = \bigcup_{i \in \mathcal{N}} T^*(i) \) forms the shortest tree rooted at \( t \). Then, for each arc \( (i, j) \in \mathcal{A} \), \( \bar{c}_{i,j} = c_{i,j} + f(T^*(j)) - f(T^*(i)) \) is the reduced cost of \( (i, j) \) relatively to \( T^* \) and \( \bar{f}(p) = \sum_{(i,j) \in p} \bar{c}_{i,j} \) is the corresponding reduced cost for a path \( p \). Now, let us recall the following result already presented in ([15]):

**Lemma 2**: Let \( T^* \) be a shortest tree rooted at \( t \) and \( \bar{c} \) the reduced cost computed for \( T^* \). Then:

1. \( 0 \leq \bar{c}_{i,j}, \forall (i, j) \in \mathcal{A} \).
2. \( \bar{c}_{i,j} = 0, \forall (i, j) \in \mathcal{A} \cap T^* \).
3. \( \bar{f}(T^*(i)) = 0, \forall i \in \mathcal{N} \).
4. \( \bar{f}(p) = f(p) + f(T^*(j)) - f(T^*(i)), \forall p \in \mathcal{P}_{i,j} \).

Now, let us remind that we denote by \( \theta(p) \) the deviation arc for the path \( p \) relatively to another path previously obtained by the procedure described above. Then, when looking for the deviation paths from \( p \) we only need to consider the vertices subsequent to the \( \text{tail}(\theta(p)) \), say \( v_k \), since the remaining ones have been already scanned. Therefore, for \( q_{v_i,j}^p \), the shortest deviation path from \( p, v_i \) is subsequent to \( v_k \) and one can state the following:

**Lemma 3**: \( \bar{f}(q_{v_i,j}^p) = \bar{f}(p) + \bar{c}_{v_i,j} \).
Proof: Note that $q_{v_i,j}^p$ is of the form $q_{v_i,j}^p = \text{sub}_p(s, v_i) \diamond (v_i, j) \diamond T^*(j)$, where $v_i \in \text{sub}_p(v_k, t) = T^*(v_k)$. Hence,

$$\bar{f}(q_{v_i,j}^p) = \bar{f}(\text{sub}_p(s, v_i)) + \bar{f}((v_i, j)) + \bar{f}(T^*(j)) = \bar{f}(p) + c_{v_i,j}. \quad \Box$$

**Corollary 1**: $\bar{f}(p) \leq \bar{f}(q_{v_i,j}^p)$.

From the previous results, it is clear that the deviation paths obtained from $p$ at node $v_i$ can be easily sorted out if $A(v_i) = \{(x, y) \in A : x = v_i\}$ is rearranged by no decreasing order of $\bar{c}$. Consequently, one needs to consider one shortest deviation path per node of $p$ from tail$(\theta(p))$ to $t$, reducing the number of elements in $D(p)$. Then, the shortest deviation path from $p$ at the node $v_i$ will be obtained using the active arc of $A(v_i)$, that is, the first arc in $A(v_i)$ that has not been used as a deviation arc.

Note that, in order to assure that all the arcs of $A(v_i)$ are considered, the first arc of $A(v_i)$ (for all $i \in N$) will be an arc of $T^*$. At last, since $T^*(s)$ is not obtained from a previous path, $\theta(T^*(s))$ is fixed as the first arc of $T^*(s)$.

The algorithm presented in Martins et al., [15], can be easily extended for the case where a $k$-uple cost is assigned to each arc of the network. That is attained by considering a total order relation in $\mathbb{R}^k$ such as the lexicographic order given in Definition 3. Therefore, $T^*$ will be formed by lexicographic shortest paths from $j$ to $t$.

From the Lemma 2, we conclude that, for any $p \in P_{s,t}$, $f(p) - \bar{f}(p)$ is a constant value and equal to $f(T^*(s)) - f(T^*(\ell))$. In consequence, as stated in the next result, the lexicographic order and the dominance relation are not affected when $c_{i,j}$ is replaced by $\bar{c}_{i,j}$.

**Lemma 4**: Let $p$ and $q$ be two paths of $P_{s,i}$. Then:

1. $f(p) \leq_L f(q) \iff \bar{f}(p) \leq_L \bar{f}(q)$;
2. $f(p) <_D f(q) \iff \bar{f}(p) <_D \bar{f}(q)$. 

Now, note that, for a vertex \( v_i \), the arcs of \( A(v_i) \) are lexicographically sorted and let \( q_{v_i,h}^p \) be a deviation path from \( p \) at \( v_i \) and using the active arc in \( A(v_i) \). If \( \text{sub}_{q_{v_i,h}^p}(s,h) \) is dominated by some subpath from \( s \) to \( h \) previously determined, then any path generated from \( q_{v_i,h}^p \) at vertex \( x \in T^*(h) \) will be also dominated. So, \( q_{v_i,h}^p \) can be ignored and a new deviation path from \( p \) at node \( v_i \) must be generated using the next arc in \( A(v_i) \).

The algorithm outlined below computes the set of nondominated paths on a network \( G \) with \( k \)-uple costs assigned to the arcs. The algorithm consists of an initialization step and an iterative cycle (step 2) for searching the elements in \( X \), the set of candidates for the next lexicographically nondominated path. In this searching process, a path in \( X \) is selected in order to create new deviation paths by using an internal cycle (step 2.2). Let us mention that in step 2.2.1, the logical function \( DT(w, \Pi_h) \) indicates when the sub-path \( w \) is dominated by some path from the set \( \Pi_h \). Finally, note that \( \Pi_h \) is the set of the temporary ND paths from \( s \) to \( h \) and, at the end of the algorithm, \( \Pi_t \) will correspond to \( \bar{D} \), the full set of nondominated paths on the network from \( s \) to \( t \).

4. Example

The network depicted in Figure 2(a) is used to illustrate the L&DP algorithm performance. The costs for the arcs are given in Table 1 and the iterative results obtained for this example are shown in Figure 3. In the graph of the figure, the successive generated deviation paths are represented with a dashed line identifying those ones not scanned so far.

The lexicographic shortest path from 1 to 6 in the network, \( T^*(s) = \langle 1, 4, 5, 6 \rangle \), is the first deviation path \( (q_1) \) to be included in \( X \). Then, \( q_1 \) is selected for scanning and, from that, two shortest deviation paths \( (q_2 \) and \( q_3 \) are generated. Note that \( q_3 \in X \) in spite of being dominated by \( q_2 \), because ND deviation paths could be obtained from \( \text{sub}_{q_3}(s,5) = \langle 1, 4, 5 \rangle \). We would like to note also that the deviation path from \( q_1 \) at the node 4, \( \langle 1, 4, 2 \rangle \) \( T^*(2) \), is discarded since \( \langle 1, 4, 2 \rangle \) is dominated by \( \langle 1, 2 \rangle \), already in \( \Pi_2 \).
L&DP algorithm

\[ \{X: \text{set of candidates for the next lexicographic nondominated path}\} \]
\[ \{\Pi_i: \text{set of temporary nondominated paths from } s \text{ to } i, i \in N\} \]
\[ \{\theta(p): \text{deviation arc of } p\} \]
\[ \{DT(p, \Pi): \text{dominance test applied to } p \text{ over } \Pi; \ DT(p, \Pi) = true \Leftrightarrow \exists q \in \Pi: q <_D p\} \]

**step 1:** (initialization)

\[ T^* \leftarrow \text{tree formed by lexicographic shortest paths from } i \text{ to } t, \forall i \in N \]
\[ \overline{c}_{ij} \leftarrow c_{ij} + f(T^*(j)) - f(T^*(i)), \forall (i, j) \in A \]
\[ \text{Sort } A(i) \text{ by non-decreasing order of } \overline{c}_{ij} \]
\[ p \leftarrow T^*(s) \quad \{\text{Assume that } p = (v_0, \ldots, v_r)\} \]
\[ \theta(p) \leftarrow (v_0, v_1) \]
\[ X \leftarrow \{p\} \]
\[ \Pi_i \leftarrow \emptyset, \forall i \in N \]

**step 2:** (searching for a new candidate to the next lexicographically ND path)

while \((X \neq \emptyset)\) do

2.1: Pick out \(p \in X\) such that \(\bar{f}(p) \leq_L \bar{f}(q), \forall q \in X\) \{Suppose \(p = (v_0, \ldots, v_r)\)\}

Assume that \(\theta(p)\) is the arc \((v_j, v_{j+1})\) of \(p\)

\[ X \leftarrow X \setminus \{p\} \]

2.2: for \(i \leftarrow j\) to \(r\) do

2.2.1: \[ w \leftarrow \text{sub}_p(s, v_i) \]

if \((DT(w, \Pi_{v_i}) = true)\) then goto step 2 \{ignore the remainder nodes of \(p\)\}
else \(\Pi_{v_i} \leftarrow \{w\} \cup \{u \in \Pi_{v_i}: DT(u, \{w\}) = false\}\):

Suppose \((v_i, v_{i+1})\) is the \(\ell\)-th arc of \(A(v_i)\), i.e., \(a_{\ell} = (v_i, v_{i+1})\)

2.2.2: \[ \text{arcFound} \leftarrow false \quad \{\text{find the next active arc in } A(v_i)\} \]

for \(x \leftarrow \ell\) to \(|A(v_i)|\) do

if \(DT(w \diamond \langle v_i, \text{head}(a_{\ell}) \rangle, \Pi_{\text{head}(a_{\ell})}) = false\)

then \(\text{arcFound} \leftarrow true\) and goto step 2.2.3
endfor \{for \(x \leftarrow \ell\) to \(|A(v_i)|\) do\}

2.2.3: \{computation of a new candidate for the next lex. ND path\}

if \((\text{arcFound} = true)\)

then \(h \leftarrow \text{head}(a_{\ell})\)

\[ q \leftarrow w \diamond \langle v_i, h \rangle \diamond T^*(h) \quad \{q^p_{v_i, h}\} \]
\[ \theta(q) \leftarrow a_{\ell} \]
\[ X \leftarrow X \cup \{q\} \]
\[ \Pi_h \leftarrow \{w \diamond \langle v_i, h \rangle\} \cup \{u \in \Pi_h : DT(u, \{w \diamond \langle v_i, h \rangle\}) = false\} \]
endfor \{for \(i \leftarrow j\) to \(r\) do\}
endwhile \{while \((X \neq \emptyset)\) do\}

**step 3:** (all the nondominated paths has been determined)

\[ D \leftarrow \Pi_i \]
A new algorithm for the MSPP

\[ (i, j) \mid (1, 2) \mid (1, 4) \mid (2, 3) \mid (2, 5) \mid (3, 4) \mid (3, 6) \mid (4, 2) \mid (4, 5) \mid (5, 3) \mid (5, 6) \]
\[ c(i, j) \mid (8.4, 1) \mid (4.7, 0) \mid (9.8, 4) \mid (8.0, 7) \mid (1.5, 9) \mid (2.6, 9) \mid (7.4, 1) \mid (3.1, 8) \mid (10.8, 2) \mid (8.2, 8) \]

Table 1: Network cost for the example of Figure 2(a).

At iteration 2, \( q_2 \) is scanned and \( q_4 \) is the only shortest deviation path to be added up to \( X \). Then, \( q_3 \) and \( q_4 \) are analysed, respectively, at iteration 3 and 4, producing no new shortest deviation paths.

5. Correctness of the algorithm

In this section, the correctness of the L&DP algorithm will be proved. Now, let us remind that \( \mathcal{P}_{s,i} \) is the set of paths from \( s \) to \( i \) and \( \bar{D}_{s,i} \subseteq \mathcal{P}_{s,i} \) corresponds to the subset of nondominated paths between those nodes. In the algorithm, the set \( \Pi_i \) is used to store the temporary ND paths from \( s \) to \( i \), that is, paths which are not discarded by the dominance test performed by the logical function \( DT(w, \Pi_i) \), that returns the value “true” if and only if there is a path \( q \in \Pi_i \) such that \( q <_D w \).

\[ (i, j) \mid (1, 4) \mid (1, 2) \mid (2, 3) \mid (2, 5) \mid (3, 6) \mid (3, 4) \mid (4, 5) \mid (4, 2) \mid (5, 6) \mid (5, 3) \]
\[ \bar{c}(i, j) \mid (0.0, 0) \mid (4.8, -2) \mid (0.0, 0) \mid (5.12, 2) \mid (0.0, 0) \mid (10.2, 16) \mid (0.0, 0) \mid (7.15, -2) \mid (0.0, 0) \mid (4.12, 3) \]

Table 2: Reduced cost associated to \( T^* \) (Figure 2(b)) where \( A(i) \) is sorted by \( \bar{c}_{i,j} \).
A new algorithm for the MSPP

<table>
<thead>
<tr>
<th></th>
<th>$X$</th>
<th>$\Pi_1$</th>
<th>$\Pi_2$</th>
<th>$\Pi_3$</th>
<th>$\Pi_4$</th>
<th>$\Pi_5$</th>
<th>$\Pi_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
<td>{$q_1$}</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>Iteration 1</td>
<td>{$q_2, q_3$}</td>
<td>{$(1)$}</td>
<td>{$(1, 2)$}</td>
<td>{$(1, 4, 5, 3)$}</td>
<td>{$(1, 4)$}</td>
<td>{$(1, 4, 5)$}</td>
<td>{$q_1$}</td>
</tr>
<tr>
<td>Iteration 2</td>
<td>{$q_3, q_4$}</td>
<td>{$(1)$}</td>
<td>{$(1, 2)$}</td>
<td>{$(1, 4, 5, 3), (1, 2, 3)$}</td>
<td>{$(1, 4)$}</td>
<td>{$(1, 4, 5), (1, 2, 5)$}</td>
<td>{$q_1, q_2$}</td>
</tr>
<tr>
<td>Iteration 3</td>
<td>{$q_4$}</td>
<td>{$(1)$}</td>
<td>{$(1, 2)$}</td>
<td>{$(1, 4, 5, 3), (1, 2, 3)$}</td>
<td>{$(1, 4)$}</td>
<td>{$(1, 4, 5), (1, 2, 5)$}</td>
<td>{$q_1, q_2$}</td>
</tr>
<tr>
<td>Iteration 4</td>
<td>$\emptyset$</td>
<td>{$(1)$}</td>
<td>{$(1, 2)$}</td>
<td>{$(1, 4, 5, 3), (1, 2, 3)$}</td>
<td>{$(1, 4)$}</td>
<td>{$(1, 4, 5), (1, 2, 5)$}</td>
<td>{$q_1, q_2, q_4$}</td>
</tr>
</tbody>
</table>

Figure 3: Simulation of the new algorithm for the example of Figure 2(a).
Later in this section, we show that, at the end of the algorithm, \( \Pi_i = \overline{D}_{s,i}, \forall i \in \mathcal{N} \).

Firstly, we recall the following result which proof can be seen in [14].

**Theorem 1** : Assume that for each cycle \( C \) of the network \( f_\ell(C) \geq 0, \forall \ell \in \{1, \ldots, k\} \).
Then, the MSPP holds the Optimality Principle stating that each nondominated path is formed by nondominated subpaths.

Now, the next Lemma 5 tells us that if a ND path is included in \( \Pi_i \), then it will never be removed from it.

**Lemma 5** : Let \( p \) be a path of \( \overline{D}_{s,i} \) that has been included in \( \Pi_i \). Then, \( p \) will never be removed from \( \Pi_i \) at any stage of the algorithm. iteration of the algorithm.

**Proof** : In fact, the removing of elements from \( \Pi_i \) uses the dominance test which always returns the value ”false” when it is applied over ND paths. \( \square \)

Let us recall that \( X \) is the set of candidates for becoming the next lexicographic shortest ND path and the algorithm stops when \( X \) turns out to be an empty set. Next, we show that the algorithm checks all the relevant arcs in the network, that is, the arcs that may lead to ND paths. Remember that the arcs of \( A(v_i) \) are ordered by non-decreasing reduced cost.

**Lemma 6** : Let \( z = \langle v_0, \ldots, v_i \rangle \) be a ND path from \( s \) to \( v_i \) and suppose that \( z \oplus T^*(v_i) \in X \). Then, all arcs of \( A(v_i) \) are scanned by the algorithm.

**Proof** : The first arc of \( A(v_i) \) is in \( T^* \) and, necessarily, is considered by the algorithm, since \( z \oplus T^*(v_i) \in X \). On the other hand, we know that this path will be selected at some iteration, on step 2, with \( \theta(p) = (v_j, v_{j+1}) \) for some \( j \in \{0, 1, \ldots, i\} \). When that happens, every vertex subsequent to \( v_j \) in \( p \), will be analyzed having in mind the generation of shortest deviation paths. In particular, \( v_i \) will be ”visited” looking for a new active arc \((v_i, h)\) of \( A(v_i) \) leading, even temporarily, to a ND path. If such an arc is found, \( q_{v_i,h}^p \) is a shortest deviation path from \( p \) included in \( X \). Later, the vertex \( v_i \) will be visited again when \( q_{v_i,h}^p \) is picked up from \( X \) and a new active arc of \( A(v_i) \) will be searched. At some point, one may not find an arc in \( A(v_i) \) that leads to a temporary ND path. This means that all the arcs in \( A(v_i) \) have been scanned by the algorithm. \( \square \)
Now, we can prove that the L&DP algorithm computes $\tilde{D}$, the full set of ND paths from $s$ to $t$ in the network.

**Theorem 2**: Let $\Pi_t$ be the set of temporary nondominated paths and assume that the Optimality Principle is verified. Then, at the end of the L&DP algorithm, $\Pi_t = \tilde{D}$.

**Proof**: Firstly, let us see that $\tilde{D} \subseteq \Pi_t$. If $\langle v_0 = s, \ldots, v_\ell = t \rangle$ is a path of $\tilde{D}_{s,t}$ then, from the Optimality Principle, all subpaths of $\langle v_0, \ldots, v_\ell \rangle$ are ND. It is easy to show, by induction, that $\langle v_0, \ldots, v_i \rangle \in \Pi_{v_i}$, for all $i \in \{0, \ldots, \ell\}$. In fact, in the first iteration of the algorithm, $\langle v_0 \rangle = \langle s \rangle$ is included in $\Pi_s$. Furthermore, if $\langle v_0, \ldots, v_i \rangle \in \Pi_{v_i}$ (for some $i \in \{0, \ldots, \ell - 1\}$) then all arcs $A(v_i)$ will be scanned (Lemma 6) and so, at some stage of the algorithm, $\langle v_0, \ldots, v_i, v_{i+1} \rangle$ will join the elements of $\Pi_{v_{i+1}}$. Since $\langle v_0, \ldots, v_i, v_{i+1} \rangle$ is a ND sub-path, from the Lemma 5, it will be not further removed from $\Pi_{v_{i+1}}$. Therefore, at the end of the algorithm, $\langle v_0 = s, \ldots, v_\ell = t \rangle$ will be an element of $\Pi_t$.

Now, let us show that $\Pi_t \subseteq \tilde{D}$. Suppose that $p \in \Pi_t$ and $p \notin \tilde{D}$. Hence, there is a path $q \in \tilde{D}$ such path $q <_D p$ and, as seen above, $q$ will be added to $\Pi_t$ at some iteration of the algorithm. This could not happen before $p$ joined $\Pi_t$ since the dominance test, performed at the step 2.2.2 of the L&DP algorithm, would prevent the possibility of including $p$ in $\Pi_t$. Therefore, $p \in \Pi_t$ when $q$ is determined by the algorithm. Then, the dominance test, in this case, made either at the step 2.2.1 or the step 2.2.3, will remove $p$ from $\Pi_t$. \[\square\]

The proof of the previous theorem shows us that it keeps valid for any node of the network, that is, when the algorithm finishes $\Pi_i = \tilde{D}_{s,i}, \forall i \in N$.

**Corollary 2**: If the Optimality Principle is verified then, at the end of the L&DP algorithm, $\Pi_i = \tilde{D}_{s,i}, \forall i \in N$.

The Theorem 2 guarantees that the L&DP algorithm determines all the ND paths from a source node to a sink node in a network. Next, we show that this is achieved in a finite number of iterations.
Lemma 7: If $0 < \lambda f(C)$, for any cycle $C$ in the network, then every temporary nondominated path has no cycles.

Proof: Let $p$ be a non simple path produced by the algorithm, that is, $p = w_1 \diamond C \diamond w_2$, where $C$ is a cycle. Then, it is enough to show that the path $w_1 \diamond w_2$ is generated earlier than $p$. In fact, let $w_1 = \langle v_0, \ldots, v_i \rangle$, $w_2 = \langle v_i, \ldots, v_k \rangle$ and $C = \langle u_0 = v_i, \ldots, u_j = v_i \rangle$. If $p$ was generated by the algorithm, then $p$ is a deviation path from a certain path $q$ previously obtained by the algorithm. Hence, $p = \text{sub}_q(s, x) \diamond \langle x, y \rangle \diamond T^*(y)$, where $(x, y) = \theta(p)$. Since $T^*(y)$ is a simple path, there is $j \in \text{sub}_q(s, x) \cap T^*(y)$ such that $C = \text{sub}_q(j, x) \diamond \langle x, y \rangle \diamond \text{sub}_{T^*(y)}(y, j)$. Consequently, $\text{sub}_q(s, j) \diamond \text{sub}_{T^*(y)}(j, t) = w_1 \diamond w_2$ is a deviation path which was formerly found by the procedure.

Theorem 3: Assuming that $0 < \lambda f(C)$ holds for every cycle $C$ in the network, the L&DP algorithm computes $\bar{D}$ in a finite number of iterations.

Proof: Under the assumption, the Optimality Principle is verified (Theorem 1). From Lemma 7, we know that the algorithm only generates elementary paths as possible elements for the sets $\Pi_i$. Therefore, only a finite number of iterations is required for, as stated in Theorem 2, computing the full set $\mathcal{D}$.

6. Computational results

Trabalho em curso

7. Conclusion

Acknowledgments

This work was partially supported by the CMUC (Centro de Matemática da Universidade de Coimbra) and the CIO/UL (Centro de Investigação Operacional da Universidade de Lisboa).

The authors would also like to thank Professor Skriver for the friendly way he made his software available.
A new algorithm for the MSPP

References


