International Journal of Foundations of Computer Science
© World Scientific Publishing Company

DEVIATION ALGORITHMS FOR RANKING SHORTEST PATHS

ERNESTO DE QUEIRÓS VIEIRA MARTINS
Centro de Informática e Sistemas da Universidade de Coimbra, Departamento de Matemática, Universidade de Coimbra, Apartado 3008, 3000 Coimbra, Portugal

MARTA MARGARIDA BRAZ PASCOAL
Centro de Informática e Sistemas da Universidade de Coimbra, Departamento de Matemática, Universidade de Coimbra, Apartado 3008, 3000 Coimbra, Portugal

and

JOSÉ LUIS ESTEVES DOS SANTOS
Centro de Informática e Sistemas da Universidade de Coimbra, Departamento de Matemática, Universidade de Coimbra, Apartado 3008, 3000 Coimbra, Portugal

Received 4 October 1998
Revised 8 February 1999
Communicated by S. Olariu

ABSTRACT

The shortest path problem is a classical network problem that has been extensively studied. The problem of determining not only the shortest path, but also listing the K shortest paths (for a given integer $K > 1$) is also a classical one but has not been studied so intensively, despite its obvious practical interest. Two different types of problems are usually considered: the unconstrained and the constrained K shortest paths problem. While in the former no restriction is considered in the definition of a path, in the constrained K shortest paths problem all the paths have to satisfy some condition – for example, to be loopless.
In this paper new algorithms are proposed for the unconstrained problem, which compute a super set of the K shortest paths. It is also shown that ranking loopless paths does not hold in general the Optimality Principle and how the proposed algorithms for the unconstrained problem can be adapted for ranking loopless paths.

Keywords: Network, tree, path, path distance, paths ranking principle, labeling.

1. Introduction

The shortest path problem is a classical network problem that has been extensively studied (see [6, 7, 8, 9, 13], among hundreds of others possible references). The problem of determining not only the shortest path, but also the second, the third, ..., the $K^{\underline{\text{th}}}$ shortest path (for a given integer $K > 1$) is also a classical one (see the classical papers, [10, 15, 27]) but has not been studied so intensively, despite its obvious practical interest; however a few hundreds of papers can be found in the literature (see the online bibliography due to Eppstein, whose WWW address is "http://liinwww.ira.uka.de/bibliography/Theory/k-path.html").

The K shortest paths problem has many real world applications, namely as a subproblem. For instance, when additional constraints are considered in a shortest path problem, paths can be ranked until the first one satisfying all the constraints is determined. Also in the multiobjective shortest path problem, [4, 5, 14, 21], the *optimal paths* can be determined by ranking paths in each one of the objectives. For details about other possible applications of the K shortest paths problem, the reader is addressed to [11].

Two classes of ranking paths problems can be considered: the unconstrained problem and the constrained one. While in the former no constraint has to be considered on the path definition, in the constrained shortest path problem all the paths have to satisfy some constraint, for example, to be loopless, [16, 18, 20], to be disjoint, [25], etc.

For the ranking of unconstrained shortest paths, the algorithms can be divided in two classes: those which are based on the Optimality Principle and those which determine a set which contains the set of the K shortest paths. However, when paths are constrained to be loopless, the Optimality Principle does not hold; so, only the algorithms in the last class can be adapted for determining the K shortest loopless paths.

This paper is divided in two parts. In the first one the unconstrained problem is studied under the following perspective – the set of the K shortest paths forms a tree in the sense of Graphs Theory, being new algorithms proposed which compute a tree containing at least all the K shortest paths. In the second part, these algorithms are adapted for ranking loopless paths.

The reader is assumed to be familiarized with the problem. Notation is the one used in previous papers, [1, 2, 3, 18, 19, 20]. Despite, some definitions and notation are introduced in the next section.

2. Definitions and Notation

Let $(\mathcal{N}, \mathcal{A})$ denote a network, where $\mathcal{N} = \{v_1, \dots, v_n\}$ (or $\mathcal{N} = \{1, \dots, n\}$ to simplify) is a finite set with n elements called nodes or vertices and $\mathcal{A} = \{a_1, \dots, a_m\}$ is a finite set with m elements called arcs or edges; moreover, each arc a_k is a pair (i, j) of nodes. When all the m arcs (i, j) are unordered pairs $(\mathcal{N}, \mathcal{A})$ is said to be an undirected network; when all the m arcs (i, j) are ordered pairs $(\mathcal{N}, \mathcal{A})$ is said to be a directed network. In what follows, with no loss of generality and unless something is explicitly said, $(\mathcal{N}, \mathcal{A})$ is a directed network.

Let s and t be two given nodes of $(\mathcal{N}, \mathcal{A})$. A path from s to t in $(\mathcal{N}, \mathcal{A})$ is an alternating sequence of nodes and arcs, of the form $\langle s = v'_1, a'_1, v'_2, \dots, a'_{r-1}, v'_r = t \rangle$, where:

1. $a'_k \in \mathcal{A}$, for any $k = 1, \dots, r-1$, and $v'_k \in \mathcal{N}$, for any $k = 1, \dots, r$;
2. $a'_k = (v'_k, v'_{k+1})$, for any $k = 1, \dots, r-1$.

Let $c_{ij} \in \mathbb{R}$ be a real number associated with arc $(i, j) \in \mathcal{A}$ and let $c(p) = \sum_{p} c_{ij}$, for any path p that can be defined between a pair of nodes of $(\mathcal{N}, \mathcal{A})$, denote the value of path p – or p path distance (or p path cost).

Let $\mathcal{P}_{ij}$ denote the set of paths that can be defined from i to j in $(\mathcal{N}, \mathcal{A})$ ($\mathcal{P}$ will be used for $\mathcal{P}_{st}$). In what follows it is assumed that $\mathcal{P}_{si} \neq \emptyset$ and $\mathcal{P}_{it} \neq \emptyset$, for any node $i \in \mathcal{N}$. Moreover, with no loss of generality, it is assumed that $r \geq 3$, for all the paths that can be defined from s to t in a given network. In fact, generality is not lost, since the given network can be enlarged by the adding of two different nodes, S and T, and two zero distance arcs, (S, s) and (t, T). There is an obvious bijection between $\mathcal{P}_{ST}$ and $\mathcal{P}_{st}$ and there is also no loss of generality when it is assumed that both the initial and terminal nodes are not repeated in a path.

A path from s to t in $(\mathcal{N}, \mathcal{A})$ is a loopless path, if all its nodes are different. A cycle is a path from some node to itself, where all the nodes are different except the first which is also the last one. A cycle C in $(\mathcal{N}, \mathcal{A})$ is said to be a negative cycle *if and only if* $\sum_{C} c_{ij} < 0$, i.e., the sum of its arcs distance is negative; C is said to be an absorbent cycle *if and only if* $\sum_{C} c_{ij} \leq 0$, i.e., the sum of its arcs distance is not positive.

In the K shortest paths problem, for a given network $(\mathcal{N}, \mathcal{A})$ and for two given nodes s and t $(s \neq t)$, it is intended to determine a set $\mathcal{P}^K = \{p_1, \ldots, p_K\} \subseteq \mathcal{P}$, such that:

1. $c(p_k) \leq c(p_{k+1})$, for any $k \in \{1, \ldots, K-1\}$;
2. $c(p_K) \leq c(p)$, for any $p \in \mathcal{P} - \mathcal{P}^K$;
3. p_k is determined just before p_{k+1}, for any $k \in \{1, \ldots, K-1\}$.

Let $p_{ij} \in \mathcal{P}_{ij}$ and $p_{j\ell} \in \mathcal{P}_{j\ell}$ be two paths defined in $(\mathcal{N}, \mathcal{A})$. Since the terminal node of p_{ij} is also the initial node of $p_{j\ell}$, the concatenation of p_{ij} and $p_{j\ell}$, denoted by $p_{ij} \diamond p_{j\ell}$, is defined as the path from i to ℓ in $(\mathcal{N}, \mathcal{A})$, which coincides with p_{ij} from i to j and with $p_{j\ell}$ from j to ℓ.

Let C be a given cycle in $(\mathcal{N}, \mathcal{A})$. For any $k \geq 1$, C^k will denote the k cycle C concatenation $\underbrace{C \diamond \ldots \diamond C}_{k\ times}$, that is, k turns around cycle C.

Let p be some path from s to t in $(\mathcal{N}, \mathcal{A})$ and let C be a given cycle in $(\mathcal{N}, \mathcal{A})$ such that its node i is also a node of p; let p_{si} and p_{it} be two paths in $(\mathcal{N}, \mathcal{A})$ such that $p = p_{si} \diamond p_{it}$. Under these assumptions, $p \diamond C^k$ will denote the path $p_{si} \diamond C^k \diamond p_{it}$, for any $k \geq 0$. (For convenience, $p \diamond C^0$ is also used to denote p).

A path is said to be finite if the number of its nodes (and arcs) is finite.

Theorem 1 *For any $i \in \mathcal{N}$, let us assume that $\mathcal{P}_{si} \neq \emptyset$ and $\mathcal{P}_{it} \neq \emptyset$. There is a finite shortest path from s to t in $(\mathcal{N}, \mathcal{A})$ if and only if there are no negative cycles in $(\mathcal{N}, \mathcal{A})$.*

Proof. Let us assume the existence of a cycle C in $(\mathcal{N}, \mathcal{A})$ such that $\sum_{C} c_{ij} < 0$ and let i be some node of C. Since, by assumption, $\mathcal{P}_{si} \neq \emptyset$ and $\mathcal{P}_{it} \neq \emptyset$, there is a path $p_{si} \in \mathcal{P}_{si}$ and a path $p_{it} \in \mathcal{P}_{it}$, such that $p = p_{si} \diamond p_{it}$ is a path from s to t. Moreover, since i is a node of path p and of cycle C, it can be concluded that $p(k) = p \diamond C^k$ is a path from s to t for any $k \geq 0$. Let $p_\infty = \lim_{k \to \infty} p(k)$. So,

$$\begin{aligned}
c(p_\infty) &= c(\lim_{k \to \infty} p(k)) \\
&= \lim_{k \to \infty} c(p(k)) \\
&= \lim_{k \to \infty} c(p \diamond C^k) \\
&= \lim_{k \to \infty} \{c(p) + c(C^k)\} \\
&= \lim_{k \to \infty} c(p) + \lim_{k \to \infty} c(C^k) \\
&= c(p) + \lim_{k \to \infty} \{k \times c(C)\} \\
&= c(p) + c(C) \times \lim_{k \to \infty} k \\
&= -\infty.
\end{aligned}$$

That is, p_∞ is a shortest path and it has a non finite number of nodes. Moreover, if q is a finite path from s to t in $(\mathcal{N}, \mathcal{A})$, then $c(q)$ is finite and q is not a shortest path. So, it may be concluded that there is no finite shortest path in $(\mathcal{N}, \mathcal{A})$.

Let us assume now the non existence of negative cycles in $(\mathcal{N}, \mathcal{A})$. In this case, since $\mathcal{P}$ is a non empty set, it is well known that there is a shortest path p which is a loopless path. So, p is a finite path. $\square$

A particular K shortest paths problem is said to be finite if there is a finite $K^{\underline{\text{th}}}$ shortest path.

Theorem 2 *Let us assume that $\mathcal{P}_{si} \neq \emptyset$ and $\mathcal{P}_{it} \neq \emptyset$, for any $i \in \mathcal{N}$. The K shortest paths problem is finite if and only if there are no negative cycles in $(\mathcal{N}, \mathcal{A})$.*
Proof. It follows directly from theorem 1. $\square$

In order to assure the finiteness of the K shortest paths problem, it is assumed the existence of no negative cycles in $(\mathcal{N}, \mathcal{A})$ throughout. Moreover, to avoid that the last computed paths are of the form $p \diamond C^k$, it is assumed that $c(C) > 0$ for any cycle C in $(\mathcal{N}, \mathcal{A})$, that is, it is assumed the existence of no absorbent cycles in $(\mathcal{N}, \mathcal{A})$.

3. The Optimality Principle

It is well known that shortest path labeling algorithms are supported by the Optimality Principle which asserts that *there is a shortest path formed by shortest sub–paths.* It could be proved that the shortest path problem satisfies the Optimality Principle if and only if *there are no negative cycles in the network.*

This principle is generalized for $K > 1$, in theorem 3.

Theorem 3 *Let us assume that $c(C) \geq 0$ for any cycle C in $(\mathcal{N}, \mathcal{A})$ and $\mathcal{P}_{si} \neq \emptyset$, $\mathcal{P}_{it} \neq \emptyset$ for any $i \in \mathcal{N}$. The $k^{\underline{\text{th}}}$ shortest path is formed by $j^{\underline{\text{th}}}$ shortest paths, for $j \leq k$.*
Proof. Let us assume that the $k^{\underline{\text{th}}}$ shortest path p_k is not formed by $j^{\underline{\text{th}}}$ shortest paths, for $j \leq k$, i.e., $p_k = p_{s\ell} \diamond p^j_{\ell h} \diamond p_{ht}$ and $p^j_{\ell h}$ is a subpath of p_k which is the $j^{\underline{\text{th}}}$ shortest path from ℓ to h and $j > k$. Since $j > k$ we conclude that there are, at least, k different paths $p^i_{\ell h}$ from ℓ to h such that $c(p^i_{\ell h}) < c(p^j_{\ell h})$, for any $1 \leq i \leq k$. But $c(p_k) = c(p_{s\ell}) + c(p^j_{\ell h}) + c(p_{ht})$ and $c(p^i_{\ell h}) < c(p^j_{\ell h})$, for any $1 \leq i \leq k$; so, as $c(p^j_{\ell h})$ is finite (theorem 2) we conclude that $c(p_{s\ell}) + c(p^i_{\ell h}) + c(p_{ht}) < c(p_k)$, for any $1 \leq i \leq k$. That is, p_k is not the $k^{\underline{\text{th}}}$ shortest path which contradicts the assumption made. $\square$

A class of ranking shortest paths algorithms is supported by theorem 3. Dreyfus, [10], was the first one to propose an algorithm in this class. This algorithm determines not only the K shortest paths between a given pair of nodes but also the K shortest paths from all nodes to t.

Yet in this class of algorithms we can consider the generalization of the labeling algorithms for the shortest path problem (Shier was the first one to propose algorithms in this subclass, [22, 23, 24]) and all the versions of the path deletion algorithm of Martins *et al.*, [1, 2, 3, 17, 19].

As it is established in theorem 4 and in opposition with what happens in the K shortest paths problem, this principle can not be generalized for $K > 1$ when paths are required to be loopless.

Theorem 4 – *It may exist a $k^{\underline{\text{th}}}$ shortest loopless path containing a $j^{\underline{\text{th}}}$ loopless subpath with $j > k$.*
Proof. To prove theorem 4 we will use the example depicted from figure 1.

From $s = 1$ to $t = 2$ there are only the following two loopless paths in the network of figure 1: $p_1 = \langle 1, (1,2), 2 \rangle$ and $p_2 = \langle 1, (1,3), 3, (3,5), 5, (5,2), 2 \rangle$, with $c(p_1) = 0$ and $c(p_2) = 2$; that is, $c(p_1) < c(p_2)$. From $s = 1$ to node 3 only the

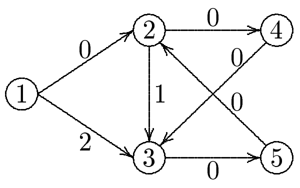

Fig. 1. Network where the OP fails for the K shortest loopless paths problem.

following three loopless paths can be computed: $q_1 = \langle 1,(1,2),2,(2,4),4,(4,3),3\rangle$, $q_2 = \langle 1,(1,2),2,(2,3),3\rangle$ and $q_3 = \langle 1,(1,3),3\rangle$, for which $c(q_1) = 0$, $c(q_2) = 1$ and $c(q_3) = 2$. So, q_3 is the third shortest loopless path from $s = 1$ to $t = 3$ and q_3 is a subpath of p_2, the second shortest loopless path from $s = 1$ to $t = 2$. $\square$

From theorem 4 it can be concluded that the Optimality Principle does not hold in general for the problem of ranking of loopless paths; as a consequence, labeling algorithms can not be generalized for $K > 1$ when only loopless paths have to be computed.

From theorem 4 it also results that all the algorithms supported by the Optimality Principle, such as Dreyfus' algorithm, all labeling algorithms due to Shier and all the versions of the path deletion algorithm can not be adapted for determining only loopless paths.

4. The Tree of the K Shortest Paths

Let $\mathcal{P}^K = \{p_1, \dots, p_K\}$ be the set of the K shortest (unconstrained or loopless) paths between a given pair of nodes and let us consider algorithm 1 which is exemplified in figure 2 for a network where $\mathcal{P}^3 = \{p_1, p_2, p_3\}$ and $p_1 = \langle 1,(1,4),4,(4,6),6\rangle$, $p_2 = \langle 1,(1,4),4,(4,1),1,(1,6),6\rangle$ and $p_3 = \langle 1,(1,6),6\rangle$.

Algorithm 1 – **Constructing a pseudo–tree, the tree of the K shortest paths.**

```
  /*        $\mathcal{T}$  —  Pseudo–tree of the $K$ shortest paths.        */

$\mathcal{T} \leftarrow \emptyset$
$k \leftarrow 1$
While $k \leq K$
do begin
        $arc \leftarrow$ first arc of $p_k$
        $head \leftarrow$ head node of $arc$
        $p \leftarrow$ subpath of $p_k$ from $s$ to $head$
        While $p \in \mathcal{T}$
        do begin
                $arc \leftarrow$ arc following $arc$ in $p_k$
                $head \leftarrow$ head node of $arc$
                $p \leftarrow$ subpath of $p_k$ from $s$ to $head$
        end
        $v_k \leftarrow$ tail node of $arc$
        $p^k_{v_k t} \leftarrow$ subpath of $p_k$ from $v_k$ to $t$
        $p^k_{s v_k} \leftarrow$ subpath of $p_k$ from $s$ to $v_k$
        $\mathcal{T} \leftarrow \mathcal{T} \cup p^k_{v_k t}$    /* In such a way that $p_k = p^k_{s v_k} \diamond p^k_{v_k t}$ is a path of $\mathcal{T}$  */
        $k \leftarrow k + 1$
end
```

For $k = 1$ the following values are successively obtained: $arc \leftarrow (1,4)$, $head \leftarrow 4$, $p \leftarrow \langle 1,(1,4),4\rangle$; since $\mathcal{T}$ is empty it can be concluded that $p \notin \mathcal{T}$ and $v_1 \leftarrow 1$,

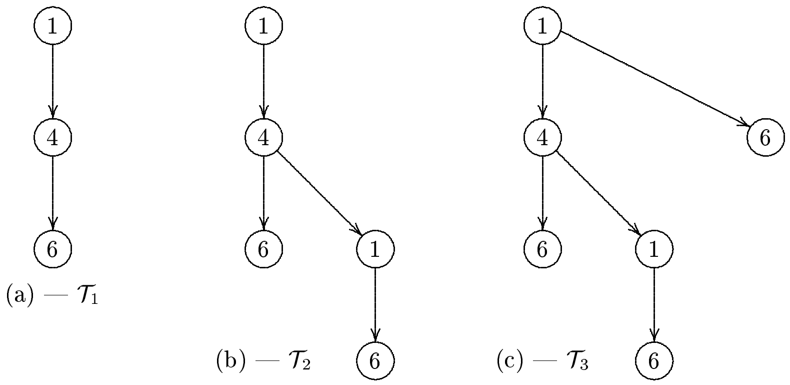

Fig. 2. The trees of shortest paths.

$p^1_{v_1 6} \leftarrow p_1,\ p^1_{1v_1} \leftarrow \emptyset,\ \mathcal{T} \leftarrow p_1, k \leftarrow 2$ – see figure 2(a).

For $k = 2$, as $p_2 = \langle 1,(1,4),4,(4,1),1,(1,6),6\rangle$, then: $arc \leftarrow (1,4)$, $head \leftarrow 4$ and $p \leftarrow \langle 1,(1,4),4\rangle$; as $\langle 1,(1,4),4\rangle \in \mathcal{T}$, we conclude that $arc \leftarrow (4,1)$, $head \leftarrow 1$ and $p \leftarrow \langle 1,(1,4),4,(4,1),1\rangle$ which is not a path of $\mathcal{T}$. That is, $v_2 \leftarrow 4$; since $p^2_{46} \leftarrow \langle 4,(4,1),1,(1,6),6\rangle$ then $p^2_{14} \leftarrow \langle 1,(1,4),4\rangle$ is added to $\mathcal{T}$ in such a way that p_2 is a path of $\mathcal{T}$ – see figure 2(b).

The algorithm continues in a similar way until $k = 3$ with $arc \leftarrow (1,6)$, $head \leftarrow 6$ and $p \leftarrow \langle 1,(1,6),6\rangle$; since p is not a path of $\mathcal{T}$ it is added to the tree – see figure 2(c).

Theorem 5 is immediate from algorithm 1.

Theorem 5 *The set $\mathcal{P}^k = \{p_1,\dots,p_k\}$ of the k shortest paths forms a pseudo-tree, for $k \geq 1$.*

For any $k \in \{1,\dots,K\}$, node v_k determined by algorithm 1 is called the deviation node of p_k; the arc of p_k whose tail node is v_k is called deviation arc of p_k and $p^k_{v_k t}$, the subpath of p_k from v_k to t, is the deviation path of p_k.

From algorithm 1 it is also immediate that nodes are repeated in $\mathcal{T}$, but we will consider all nodes as being different; so, $\mathcal{T}$ is a pseudo-tree – called the tree of the K shortest paths. The objective of any algorithm for ranking the K shortest paths is to construct $\mathcal{T}$; to achieve this, a pseudo-tree which contains $\mathcal{T}$ is usually constructed. So, a class of algorithms can be designed for which a set of candidates to p_k is used, being p_k the shortest path in this set. These algorithms are supported by theorems 5 and 6.

Theorem 6 *For $k \geq 1$, let $\mathcal{T}_k$ denote the tree of the k shortest paths and let $p^{k+1}_{v_{k+1}t}$ be the deviation path of p_{k+1}. Then $p^{k+1}_{v_{k+1}t}$ is the shortest path from v_{k+1} to t whose first arc is not an arc of $\mathcal{T}_k$ with v_{k+1} as tail node.*

Proof. Let us assume that $p^{k+1}_{v_{k+1}t}$, the deviation path of p_{k+1}, is not the shortest path from v_{k+1} to t whose first arc is not an arc of $\mathcal{T}_k$ with v_{k+1} as tail node. That is, there is a path q from v_{k+1} to t whose first arc is not an arc of $\mathcal{T}_k$ with v_{k+1} as tail node, such that $\sum_{q} c_{ij} < \sum_{p^{k+1}_{v_{k+1}t}} c_{ij}$. Since v_{k+1} is the tail node of the first arc of q and this arc is not an arc of $\mathcal{T}_k$, it can be concluded that $q \notin \mathcal{T}_k$. That is, p_{k+1} is not the $(k+1)$th shortest path which contradicts the hypothesis. Since paths from v_{k+1} to t whose first arc is an arc of $\mathcal{T}_k$ with v_{k+1} as tail node have been already considered, the theorem follows. □

From algorithm 1 it is also immediate that $\mathcal{P}^k$ can be formed only by loopless paths. That is, the K shortest loopless paths form a pseudo–tree – the tree of the K shortest loopless paths.

The concepts of deviation node and deviation path of the $k^{\underline{\text{th}}}$ shortest loopless path are similar to the same concepts for the unconstrained problem.

Theorem 7 – *Theorem 6 is still valid when $\mathcal{P}^k$ is constrained to the k shortest loopless paths between a given pair of nodes.*
Proof. Similar to the proof of theorem 6 □

Theorems 5 and 7 are the support of a class of algorithms to determine the K shortest loopless paths. The best known algorithm in this class is due to Yen, [27]; this algorithm will be briefly reviewed in paragraph 6.

5. The Unconstrained Paths Problem

In this section two new algorithms are presented for the ranking unconstrained paths problem. The first one can be viewed as a generalization of Yen's algorithm for ranking loopless paths.

5.1. Generalization of Yen's algorithm

In this subsection $\mathcal{T}_k$ will denote a pseudo-tree which contains the tree of the k shortest paths.

For any node $v \in \mathcal{N}$ let $\mathcal{A}(v)$ denote the set of arcs of $(\mathcal{N}, \mathcal{A})$ whose tail node is v; for any $k \geq 1$, let $\mathcal{A}_{\mathcal{T}_k}(v)$ denote the set of arcs of $\mathcal{T}_k$ whose tail node is v, a node of $p^k_{v_k t}$ – the deviation path of p_k.

In the algorithm to be described in this section, a set X of paths candidates to the following shortest one is used. Obviously, X is initialized with the shortest path p_1.

In the $k^{\underline{\text{th}}}$ step of the algorithm, the $k^{\underline{\text{th}}}$ shortest path is picked out from X and some new candidates to the $(k+1)^{\underline{\text{th}}}$ shortest path are generated. Since p_k coincides with some p_ℓ, for $1 \leq \ell < k$, from the initial node s until the deviation node v_k of p_k, all nodes until v_k were already considered in the attempt of generating new candidates. So, from all the nodes v of $p^k_{v_k t}$, the shortest path $p^\star_{vt}$, from v to t whose first arc is not an arc of $\mathcal{A}_{\mathcal{T}_k}(v)$, will be computed. From theorem 6, $p^k_{sv} \diamond p^\star_{vt}$ is a new candidate to p_{k+1}. $p^\star_{vt}$ can be easily determined since $\mathcal{T}^\star_t$, the tree of the shortest paths from any node $i \in \mathcal{N}$ to t, had been previously computed. In fact, the arc $(v, x) \in \mathcal{A}(v) - \mathcal{A}_{\mathcal{T}_k}(v)$ has to be determined in such a way that $c_{vx} + c(p^\star_{xt})$ is minimized, where $p^\star_{xt}$ stands for the shortest path from x to t, that is, the path which is determined in $\mathcal{T}^\star_t$ from x to t. It must be noticed that $\mathcal{A}(v) - \mathcal{A}_{\mathcal{T}_k}(v)$ can be the empty set, which means that all deviations paths for node v were already determined. Moreover, the deviation node of some path p_k, for $k \geq 1$ can be the initial node itself, which implies that all nodes of p_k have to be considered in the attempt for generating candidates to p_{k+1}.

This algorithm works similarly to Yen's algorithm for ranking loopless paths, [27]. The only difference consists in the path definition; that is, while in Yen's algorithm only loopless paths are computed, in the proposed algorithm paths may eventually have loops. That is the reason why the algorithm is considered a generalization of Yen's algorithm.

It must be also noticed that $\mathcal{T}^\star_t$ can be easily determined with some labeling algorithm for the classical shortest path problem being enough to reverse the orientation of all the arcs and to consider t as the initial node.

The algorithm is exemplified with the network in figure 3, whose tree of the shortest paths from $i \in \mathcal{N}$ to t is represented in figure 4.

So, $p_1 = \langle 1, (1,4), 4, (4,6), 6 \rangle$ is the shortest path from s to t and p_1 is a path of $\mathcal{T}^\star_t$; moreover, v_1, its deviation node, is the initial node $s = 1$. That is, $X = \{p_1\}$

Algorithm 2 – **Generalization of Yen's algorithm**

```
/*        $X$   —   Set of candidates to the following shortest path.        */

Compute $\mathcal{T}_t^\star$
$p_1 \leftarrow$ shortest path from $s$ to $t$          /*   $p_1$ is a path of $\mathcal{T}_t^\star$   */
$k \leftarrow 1$
$X \leftarrow \{p_k\}$
$\mathcal{T}_k \leftarrow \{p_k\}$
While $k < K$ and $X \neq \emptyset$
do begin
    $X \leftarrow X - \{p_k\}$
    $v_k \leftarrow$ deviation node of $p_k$
    for each node $v \in p^k_{v_k t}$
    do begin
        if $\mathcal{A}(v) - \mathcal{A}_{\mathcal{T}_k}(v) \neq \emptyset$
        then begin
            Compute the arc $(v,x)$ such that $c_{vx} + c(p^\star_{xt})$
                is minimized over $\mathcal{A}(v) - \mathcal{A}_{\mathcal{T}_k}(v)$
                /*  $p^\star_{xt}$ is a path of $\mathcal{T}_t^\star$  */
            $q \longleftarrow p^k_{sv} \diamond \langle v, (v,x), x\rangle \diamond p^\star_{xt}$
            $X \longleftarrow X \cup \{q\}$
            $q_{vt} \longleftarrow \langle v, (v,x), x\rangle \diamond p^\star_{xt}$
            $\mathcal{T}_k \longleftarrow \mathcal{T}_k \cup \{q_{vt}\}$
                /*  In such a way that $p^k_{sv} \diamond q_{vt}$ is a path of $\mathcal{T}_k$  */
        end
    end
    $k \leftarrow k + 1$
    $p_k \leftarrow$ shortest path in $X$
end
```

and $\mathcal{T}_1$ is path p_1 itself – see figure 5. Path q_2 is generated from the deviation node of p_1 and added to $\mathcal{T}_1$. In fact, $v_1 = 1$ is the tail node of arcs $(1,2), (1,3)$ and $(1,4)$; since $(1,4)$ is an arc of $\mathcal{T}_1$ with $v_1 = 1$ as tail node, the minimum of $\{c_{12} + c(p^\star_{26}), c_{13} + c(p^\star_{36})\} = \{0+3, 0+2\}$ has to be computed. This minimum is attained for for $\{c_{13} + c(p^\star_{36})\}$ – so, $\langle 1, (1,3), 3\rangle \diamond p^\star_{36}$ is added to $\mathcal{T}_1$.

Similarly for nodes 4 and 6, to obtain $\mathcal{T}_2$ – see figure 5. Notice that no path is added to $\mathcal{T}_1$ from node 6 since $\mathcal{A}(6) = \emptyset$ (which implies that $\mathcal{A}_{\mathcal{T}}(6) = \emptyset$, for any tree $\mathcal{T}$).

The computational complexity of Yen's generalization algorithm is $\mathcal{O}(Km)$, after the determination of $\mathcal{T}_t^\star$ and when the worst case analysis is considered. In fact, in the worst case, no more than n different nodes have to be considered after the deviation node, when new candidate paths are being added to the set X. That is, for each k all the set of arcs is analyzed in the worst case.

5.2. Reduced Costs

Let $\mathcal{T}_t$ be some tree rooted at t, that is, a tree where there is a (unique) path from any node $i \in \mathcal{N}$ to t; for any node $i \in \mathcal{N}$, let $\pi_i(\mathcal{T}_t)$ denote the cost of p_{it} – the path from i to t in $\mathcal{T}_t$. For any arc $(i,j) \in \mathcal{A}$ let $\bar{c}_{ij} = \pi_j(\mathcal{T}_t) - \pi_i(\mathcal{T}_t) + c_{ij}$ be the reduced cost of arc $(i,j) \in \mathcal{A}$ associated with $\mathcal{T}_t$.

Theorem 8 plays an important role in ranking paths algorithms.

Theorem 8 *For any two nodes $x, y \in \mathcal{N}$ such that $x \neq y$, let $p, q \in \mathcal{P}_{xy}$ be two paths from x to y. For any tree rooted at t, $\mathcal{T}_t$:*

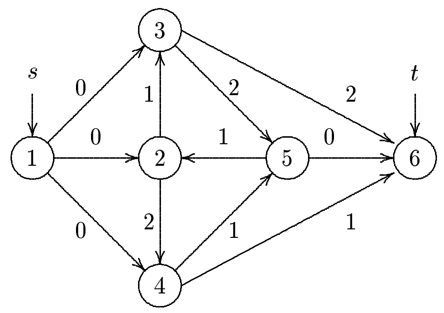

Fig. 3. Network to exemplify algorithms.

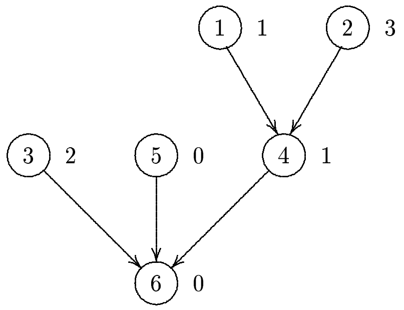

Fig. 4. $\mathcal{T}_t^\star$ for the network in Fig. 3.

1 – $c(p) = \sum_p c_{ij} = \sum_q c_{ij} = c(q)$ *if and only if* $\bar{c}(p) = \sum_p \bar{c}_{ij} = \sum_q \bar{c}_{ij} = \bar{c}(q)$.

2 – $c(p) = \sum_p c_{ij} < \sum_q c_{ij} = c(q)$ *if and only if* $\bar{c}(p) = \sum_p \bar{c}_{ij} < \sum_q \bar{c}_{ij} = \bar{c}(q)$.

Proof. The proof follows immediate from the definition of reduced cost. In fact, from $\bar{c}(p) = \sum_p \bar{c}_{ij} = \sum_q \bar{c}_{ij} = \bar{c}(q)$ it can be successively concluded:

$$\begin{aligned} \sum_p (\pi_j(\mathcal{T}_t) - \pi_i(\mathcal{T}_t) + c_{ij}) &= \sum_q (\pi_j(\mathcal{T}_t) - \pi_i(\mathcal{T}_t) + c_{ij}), \\ \pi_y(\mathcal{T}_t) - \pi_x(\mathcal{T}_t) + \sum_p c_{ij} &= \pi_y(\mathcal{T}_t) - \pi_x(\mathcal{T}_t) + \sum_q c_{ij}, \\ \sum_p c_{ij} &= \sum_q c_{ij}. \end{aligned}$$

That is, $c(p) = c(q)$.

The proof is similar for statement 2. □

From theorem 8 it can be concluded that ranking paths may be achieved either using the cost $c(p_k)$ (and c_{ij}, for every arc $(i,j) \in \mathcal{A}$) or the reduced cost $\bar{c}(p_k)$ (and $\bar{c}_{ij}$, for every arc $(i,j) \in \mathcal{A}$). The following theorem shows the advantage of using reduced costs.

Theorem 9 *Let $\mathcal{T}_t^\star$ be the shortest tree rooted at t, that is, the tree of shortest paths from every node $i \in \mathcal{N}$ to t. Then, $\bar{c}_{ij} \geq 0$ for every arc $(i,j) \in \mathcal{A}$; moreover, $\bar{c}_{ij} = 0$ for every arc $(i,j) \in \mathcal{A} \cap \mathcal{T}_t^\star$, that is, for any arc in the shortest tree rooted at t.*

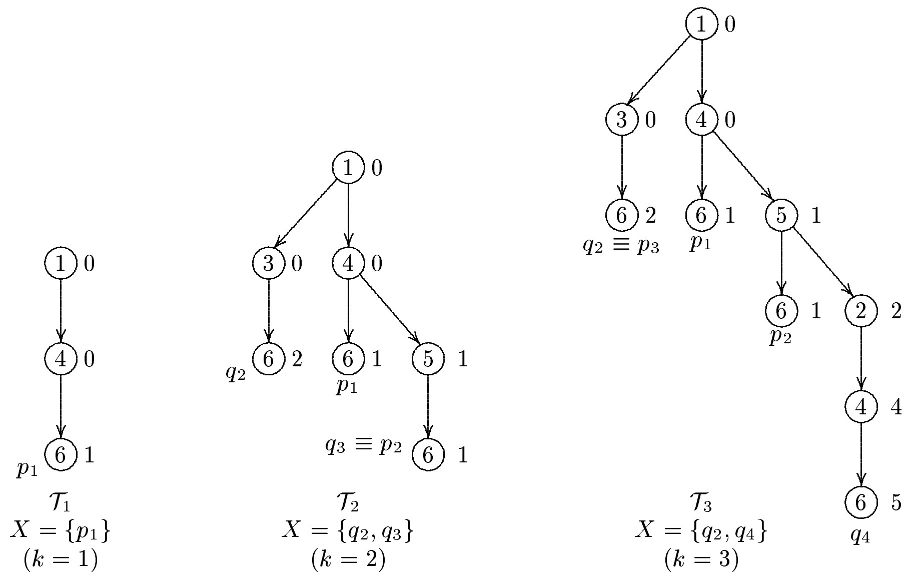

Fig. 5. The first three steps of the generalization of Yen's algorithm.

Proof. Let $(i, j) \in \mathcal{A}$ be some arc in $(\mathcal{N}, \mathcal{A})$ and let us assume that $\bar{c}_{ij} < 0$. Then, by definition, $\bar{c}_{ij} = \pi_j - \pi_i + c_{ij}$ where π_k stands for $\pi_k(\mathcal{T}_t^\star)$, for any node $k \in \mathcal{N}$. It can be concluded that $\langle i, (i, j), j\rangle \diamond p_{jt}^\star$ is a path from i to t. Moreover, $c(\langle i, (i, j), j\rangle \diamond p_{jt}^\star) = c_{ij} + c(p_{jt}^\star)$; that is, $c(\langle i, (i, j), j\rangle \diamond p_{jt}^\star) = c_{ij} + \pi_j$. Since $\bar{c}_{ij} = \pi_j - \pi_i + c_{ij} < 0$ then $c_{ij} + \pi_j < \pi_i$, that is, there is a path from i to t whose cost is less than the cost of the shortest path which is impossible. So, $\bar{c}_{ij} \geq 0$. If $(i, j) \in \mathcal{A} \cap \mathcal{T}_t^\star$, π_i is the cost of the shortest path $p_{it}^\star$ from i to t which is determined in $\mathcal{T}_t^\star$ and $p_{it}^\star = \langle i, (i, j), j\rangle \diamond p_{jt}^\star$. So, $\pi_i = c_{ij} + \pi_j$ and $\pi_j - \pi_i + c_{ij} = \bar{c}_{ij} = 0$. □

The advantage of using the reduced costs associated to the shortest tree rooted at t in ranking shortest paths algorithms is immediate from theorem 9. For instance, in the generalization of Yen's algorithm more efficiency is achieved in the determination of the arc $(v, x) \in \mathcal{A}(v) - \mathcal{A}_{\mathcal{T}_k}(v)$ which minimizes $\bar{c}_{vx} + \bar{c}(p_{xt}^\star)$, since $p_{xt}^\star$ is a path of $\mathcal{T}_t^\star$ and $\bar{c}(p_{xt}^\star) = 0$ for any node $x \in \mathcal{N}$. That is, in the $k^{\underline{\text{th}}}$ step, it is enough to determine the arc $(v, x) \in \mathcal{A}(v) - \mathcal{A}_{\mathcal{T}_k}(v)$ which minimizes $\bar{c}_{vx}$, for each node v in the deviation path of p_k.

Eppstein was the first one to take advantage of using reduced costs associated to the shortest tree rooted at t, [11].

In next subsection an algorithm is presented that can be viewed as a Yen's generalization algorithm improvement.

5.3. MPS *algorithm*

In section 5.2 it was shown how the efficiency of Yen's generalization algorithm can be improved using reduced costs. However, more efficiency can yet be achieved when a suitable data structure is used to define $(\mathcal{N}, \mathcal{A})$. In fact, in section 5.2 it was proved that, in the $k^{\underline{\text{th}}}$ step of Yen's generalization algorithm, it is enough to determine the arc $(v, x) \in \mathcal{A}(v) - \mathcal{A}_{\mathcal{T}_k}(v)$ which minimizes $\bar{c}_{vx}$, for each node v in the deviation path of p_k. If the set of arcs is grouped for each tail node, and for each set $\mathcal{A}(v)$ arcs are sorted by their reduced costs, the determination of arc

$(v,x) \in \mathcal{A}(v) - \mathcal{A}_{\mathcal{T}_k}(v)$ which minimizes $\bar{c}_{vx}$ is immediate – (v,x) it is the first arc in the set $\mathcal{A}(v) - \mathcal{A}_{\mathcal{T}_k}(v)$.

In the next subsection a data structure for representing $(\mathcal{N}, \mathcal{A})$ is defined which allows more efficiency for the algorithm.

5.3.1. Sorted forward star form

Let us arrange $\mathcal{A} = \{a_1, \ldots, a_m\}$ in such a way that $k < \ell$ if and only if the tail node of a_k is not greater than the tail node of a_ℓ; that is, $\mathcal{A}$ is arranged in such a way that $\mathcal{A} = \mathcal{A}(1) \cup \ldots \cup \mathcal{A}(n)$, where n is the number of nodes of $(\mathcal{N}, \mathcal{A})$. Moreover, for every $k \in \{1, \ldots, n\}$, the set $\mathcal{A}(k)$ is sorted by the reduced cost of their arcs; that is, if $\mathcal{A}(k) = \{\hat{a}_1, \ldots, \hat{a}_\ell\}$ then $\bar{c}(\hat{a}_1) \leq \ldots \leq \bar{c}(\hat{a}_\ell)$.

Notice that rearranging $\mathcal{A}$ in the sorted forward star form can be done with $\mathcal{O}(m \log n)$ operations. In fact, let m_i denote the number of arcs whose tail node is i; so, n sorts have to be executed to rearrange the network in the sorted forward star form. A worst case analysis for an heap sorting requires $\mathcal{O}(m_i \log m_i)$ time for a m_i–array, [26]. That is, sorting the forward star form can be done in $\mathcal{O}(\sum_{i=1}^{n} m_i \log m_i)$ time, or $\mathcal{O}(m \log n)$ since $m_i < n$, for any $i \in \{1, \ldots, n\}$.

More details about the sorted forward star form can be found in [8].

5.3.2. The algorithm

MPS algorithm is stated in algorithm 3 and exemplified with the network in figure 3.

The reduced costs associated to the shortest tree rooted at t – see figure 4 – are shown in table 1.

$(i,j) \in \mathcal{A}$	π_i	π_j	c_{ij}	$\bar{c}_{ij} = \pi_j - \pi_i + c_{ij}$
$(1,2)$	1	3	0	2
$(1,3)$	1	2	0	1
$(1,4)$	1	1	0	0
$(2,3)$	3	2	1	0
$(2,4)$	3	1	2	0
$(3,5)$	2	0	2	0
$(3,6)$	2	0	2	0
$(4,5)$	1	0	1	0
$(4,6)$	1	0	1	0
$(5,2)$	0	3	1	4
$(5,6)$	0	0	0	0

Table 1. Reduced costs associated to the shortest tree rooted at t of Fig. 4.

According to these reduced costs and after rearranging $(\mathcal{N}, \mathcal{A})$ in the sorted forward star form:

$$\mathcal{A}(1) = \{(1,4),(1,3),(1,2)\}, \mathcal{A}(2) = \{(2,3),(2,4)\}, \mathcal{A}(3) = \{(3,5),(3,6)\},$$
$$\mathcal{A}(4) = \{(4,5),(4,6)\}, \mathcal{A}(5) = \{(5,6),(5,2)\} \text{ and } \mathcal{A}(6) = \emptyset.$$

From now on, all steps of MPS algorithm are coincident with those of Yen's generalization algorithm. However, the determination of new candidate paths results easier. Notice, for instance, that for generating candidates from path p_2, nodes $4, 5$ and 6 have to be scanned; while for node $v = 4$, $\mathcal{A}(4) = \mathcal{A}_{\mathcal{T}_2}(4) = \{(4,5),(4,6)\}$, so $\mathcal{A}(4) - \mathcal{A}_{\mathcal{T}_2}(4) = \emptyset$, the first arc of $\mathcal{A}(5) - \mathcal{A}_{\mathcal{T}_2}(5)$ is arc $(5,2)$ and $\mathcal{A}(6)$ is the empty set – see $\mathcal{T}_2$ and $\mathcal{T}_3$ in figure 5.

Algorithm 3 – **Algorithm MPS**

```
/*        $X$   —   Set of candidates to the following shortest path.        */

Compute $\mathcal{T}_t^\star$
Compute $\bar{c}_{ij}$, for any arc $(i,j) \in \mathcal{A}$
Rearrange the set of arcs of $(\mathcal{N}, \mathcal{A})$ in the sorted forward star form
        /*      For the computed reduced costs      */
$p_1 \leftarrow$ shortest path from $s$ to $t$        /*   $p_1$ is a path of $\mathcal{T}_t^\star$   */
$k \leftarrow 1$
$X \leftarrow \{p_k\}$
$\mathcal{T}_k \leftarrow \{p_k\}$
While $k < K$ and $X \neq \emptyset$
do begin
    $X \leftarrow X - \{p_k\}$
    $v_k \leftarrow$ deviation node of $p_k$
    for each node $v \in p^k_{v_k t}$
    do begin
        if $\mathcal{A}(v) - \mathcal{A}_{\mathcal{T}_k}(v) \neq \emptyset$
        then begin
            $(v,x) \leftarrow$ first arc in the set $\mathcal{A}(v) - \mathcal{A}_{\mathcal{T}_k}(v)$
            $q \longleftarrow p^k_{sv} \diamond \langle v,(v,x),x \rangle \diamond p^\star_{xt}$
            $X \longleftarrow X \cup \{q\}$
            $q_{vt} \longleftarrow \langle v,(v,x),x \rangle \diamond p^\star_{xt}$
            $\mathcal{T}_k \longleftarrow \mathcal{T}_k \cup \{q_{vt}\}$
              /*  In such a way that $p^k_{sv} \diamond q_{vt}$ is a path of $\mathcal{T}_k$  */
        end
    end
    $k \leftarrow k + 1$
    $p_k \leftarrow$ shortest path in $X$
end
```

The computational complexity of MPS algorithm is $\mathcal{O}(m \log n + Kn)$, when a worst case analysis is considered. In fact, the determination of $\mathcal{T}_t^\star$ can be achieved in $\mathcal{O}(m + n \log n)$ time using Fibonacci heaps, [12]; computing the arcs reduced cost is done in $\mathcal{O}(m)$ and sorting the forward star form can be done in $\mathcal{O}(m \ \log n)$. So, the $\mathcal{T}_t^\star$ determination, plus the reduced costs computation and sorting the forward star form can be achieved in $\mathcal{O}(m + n \log n + m + m \log n)$, that is, in $\mathcal{O}(m \log n)$ time. Moreover, in the worst case no more than n different nodes have to be considered after the deviation node, when new candidate paths are being added to the set X. So, for each k, n new candidate paths can be generated which can be achieved in $\mathcal{O}(n)$ and MPS algorithms ranks the K shortest paths in $\mathcal{O}(m \log n + Kn)$ time, considering a worst case analysis.

6. The Loopless Path Problem

In this section the ranking of loopless paths is considered. The classical Yen's algorithm is briefly reviewed and the adaptation for ranking loopless paths of the MPS algorithm is also presented.

6.1. Yen's algorithm

Like all the algorithms in this class, Yen's algorithm uses a set X of paths each of one candidate to the following loopless path. In the $k^{\underline{\text{th}}}$ step of the algorithm, the $k^{\underline{\text{th}}}$ shortest loopless path is picked out from X – actually, it is the shortest path in X and it is a loopless path too – and some new candidates to the $(k+1)^{\underline{\text{th}}}$ shortest loopless path are generated and added to X. Obviously, X is initialized with p_1, the shortest loopless path.

Let p_k be the $k^{\underline{\text{th}}}$ shortest loopless path, just picked out from X; let v_k and p^k_{uv} denote the deviation node of p_k and the subpath of p_k from node u to node v, respectively. So, for each node v of path $p^k_{v_k t}$ the shortest loopless path from v to t, whose first arc is not an arc of $\mathcal{T}_k$ with v as tail node and such that $p^k_{sv} \diamond p^k_{vt}$ is a loopless path, will be determined. This is easily achieved by removing from the network all the nodes of p^k_{sv}, except node v, so as the arcs of $\mathcal{T}_k$ with v as tail node and determining the shortest path in the resulting network. Notice that the arcs whose tail node or/and head node had been removed, can be removed too.

The nodes of p^k_{sv} removal assures that the shortest path in the resulting network is a loopless path; the removal of arcs of $\mathcal{T}_k$ with v as tail node assures that the shortest path of the resulting network had never been determined before.

It must be noticed that for each k no more that n new candidate paths are generated. So, Yen's algorithm solves Kn shortest path problems at most. For undirected networks, Katoh, Ibaraki and Mine proposed a specialization of Yen's algorithm which allows the determination of, at most, three new candidate paths for each loopless path p_k, [16].

6.2. Adaptation for loopless paths of algorithms in a subclass

The algorithms for ranking unconstrained paths which are being considered in this paper, can eventually be adapted for ranking loopless paths; since this adaptation is similar for both, only MPS adaptation will be (briefly) described.

Let us assume that p_k, the $k^{\underline{\text{th}}}$ shortest loopless path was just picked out (and removed) from the set X of candidates to the $(k+1)^{\underline{\text{th}}}$ shortest loopless path. In general, some new candidates to p_{k+1} will be generated from p_k; to achieve this, for each node v of p^k_{vt} – the subpath of p_k from v to t – the shortest path $p^\star_{vt}$ from v to t, whose first arc is not an arc of $\mathcal{T}_k$ with v as tail node, will be computed if some path exists under this condition.

Let us assume that $\bar{a}_\ell$ is the p_k arc whose tail node is v. Let us assume too that $tail(\bar{a}_{\ell+1}) = v$ and $head(\bar{a}_{\ell+1}) = u$; that is, $\bar{a}_{\ell+1} = (v, u)$. Once arcs are arranged in the sorted forward star form there exists a path from v to t whose first arc is not an arc of $\mathcal{T}_k$ if and only if $tail(\bar{a}_{\ell+1}) = v$, and $p^\star_{vt}$ is easily computed; in fact, if $p_{ut}(\mathcal{T}_t)$ stands for the path from u to t which is determined in $\mathcal{T}_t$, then $p^\star_{vt} = \langle v, (v,u), u\rangle \diamond p_{ut}(\mathcal{T}_t)$. This conclusion results from the fact of the reduced cost of path $p_{ut}(\mathcal{T}_t)$ being zero, [18, 20]. So, for each node v of $p^k_{v_k t}$, X is updated with $p^k_{sv} \diamond p^\star_{vt}$, if such a path exists.

However, the two loopless paths concatenation is not necessarily a loopless path. As a consequence, $p^\star_{vt} = \langle v, (v,u), u\rangle \diamond p_{ut}(\mathcal{T}_t)$ can be a path with loops and all the paths in the set X are not necessarily loopless. To reduce the number of non loopless paths in X, the arc $\bar{a}_{\ell+r}$ can be determined for each node v of $p^k_{v_k t}$, such as:

- $\bar{a}_\ell$ is the p_k arc whose tail node is v,
- $tail(\bar{a}_{\ell+r}) = v$, (and $head(\bar{a}_{\ell+r}) = u$)
- $p^k_{sv} \diamond \langle v, \bar{a}_{\ell+r}, u\rangle$ is a loopless path and $p^k_{sv} \diamond \langle v, \bar{a}_{\ell+i}, u\rangle$ is not a loopless path for $i \in \{1, \ldots, r-1\}$,

that is, $\bar{a}_{\ell+r}$ is the first arc following $\bar{a}_\ell$ in the sorted forward star form whose tail node is yet node v and whose concatenation with p^k_{sv} is still a loopless path.

Furthermore, if a path p_k picked out (and removed) from X is not a loopless path, for generating candidates to the $k^{\underline{\text{th}}}$ shortest loopless path (note that p_k is not loopless, so it is not the $k^{\underline{\text{th}}}$ shortest loopless path) the nodes v from its subpath

$p^k_{v_k t}$ have to be analyzed while the two paths concatenation $p^k_{sv_k} \diamond p^k_{v_k v}$ is a loopless path, [18, 20].

Notice that an upper bound for the number of candidate paths picked out from X can not be established. However, in practice the algorithm seems to perform well, [18, 20].

7. Conclusion

A class of algorithms for the ranking of shortest paths problem was presented in this paper. These algorithms determine the optimal solution using a set of candidates to the following path. Obviously, its efficiency depends on the number of candidates in this set; that is, algorithms will be more efficient if less paths are generated.

For the unconstrained problem Eppstein's algorithm is the one which presents the best theoretical computational complexity, followed by MPS algorithm. However, in practice it appears that MS algorithm performs better, [19, 20], but MPS algorithm appears to be a good alternative. A more or less exhaustive computational work about all the algorithms is needed to confirm, or not, this assumption. This will be the subject of a future paper.

One great advantage of the algorithms in this class is the possibility of being adapted for ranking loopless paths without loosing its practical efficiency, [18, 20]. This adaptation is also presented for MPS algorithm.

We must point out that all the algorithms presented (or referred) in the paper, can be used either for directed or undirected networks. In fact, undirected arcs can be split in two opposite direction arcs whose cost is the one of the original undirected arc.

Acknowledgements

Research developed within CISUC ("Centro de Informática e Sistemas da Universidade de Coimbra") and partly supported by the portuguese Ministry of Science and Technology (MCT), under PRAXIS XXI Project of JNICT ("Junta Nacional de Investigação Científica e Tecnológica").

The authors are in debt to the referees for their helpful comments.

References

1. J.A. Azevedo, M.E.O.S. Costa, J.J.E.R.S. Madeira, and E.Q.V. Martins. An algorithm for the ranking of shortest paths. *European Journal of Operational Research*, 69:97–106, 1993.
2. J.A. Azevedo, J.J.E.R.S. Madeira, E.Q.V. Martins, and F.M.A. Pires. A shortest paths ranking algorithm, 1990. Proceedings of the Annual Conference AIRO'90, Models and Methods for Decision Support, Operational Research Society of Italy, 1001-1011.
(http://www.mat.uc.pt/~eqvm/cientificos/investigacao/Artigos/K1.ps.gz)
3. J.A. Azevedo, J.J.E.R.S. Madeira, E.Q.V. Martins, and F.M.A. Pires. A computational improvement for a shortest paths ranking algorithm. *European Journal of Operational Research*, 73:188–191, 1994.
4. J.C.N. Clímaco and E.Q.V.M. Martins. On the determination of the nondominated paths in a multiobjective network problem. Proceedings of V Sympösium über Operations Research, Köln , (1980), in *Methods in Operations Research*, 40, (Anton Hain, Königstein , 1981), 255–258.
(http://www.mat.uc.pt/~eqvm/cientificos/investigacao/Artigos/CM.ps.gz)
5. J.C.N. Clímaco and E.Q.V.M. Martins. A bicriterion shortest path algorithm. *European Journal of Operational Research*, 11:399–404, 1982.

6. E.V. Denardo and B.L. Fox. Shortest route methods: reaching, pruning and buckects. *Operations Research*, 27:161–186, 1979.

7. N. Deo and C. Pang. Shortest paths algorithms: taxonomy and annotation. *Networks*, 9:275–323, 1979.

8. R. Dial, G. Glover, D. Karney, and D. Klingman. A computational analysis of alternative algorithms and labelling techniques for finding shortest path trees. *Networks*, 9:215–348, 1979.

9. E. Dijkstra. A note on two problems in connection with graphs. *Numerical Mathematics*, 1:395–412, 1959.

10. S.E. Dreyfus. An appraisal of some shortest-path algorithms. *Operations Research*, 17:395–412, 1969.

11. D. Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28:652–673, 1998.

12. M.L. Fredman and R.E. Tarjan. Fibonacci heaps and their uses in improving network optimization algorithms. *Journal of the Association for Computing Machinery*, 34:596–615, 1987.

13. G. Gallo and S. Pallotino. Shortest path methods: a unifying approach. *Mathematical Programming Study*, 26:38–64, 1986.

14. P. Hansen. Bicriterion path problems. in *Multiple Criteria Decision Making: Theory and Applications*, editors: G. Fandel and T. Gal, Lectures Notes in Economics and Mathematical Systems, 177, 109-127, Springer Heidelberg, 1980.

15. R. Hoffman and R.R. Pavley. A method for the solution of the n–th best path problem. *Journal of the Association for Computing Machinery*, 6:506–514, 1959.

16. N. Katoh, T. Ibaraki, and H. Mine. An efficient algorithm for k shortest simple paths. *Networks*, 12:411–427, 1982.

17. E.Q.V. Martins. An algorithm for ranking paths that may contain cycles. *European Journal of Operational Research*, 18:123–130, 1984.

18. E.Q.V. Martins, M.M.B. Pascoal, and J.L.E. Santos. A new algorithm for ranking loopless paths. Submitted, 1997.
(http://www.mat.uc.pt/∼eqvm/cientificos/investigacao/Artigos/mps.ps.gz)

19. E.Q.V. Martins and J.L.E. Santos. A new shortest paths ranking algorithm. Submitted, 1996.
(http://www.mat.uc.pt/∼eqvm/cientificos/investigacao/Artigos/K.ps.gz)

20. M.M.B. Pascoal. Algoritmos para a enumeração dos k trajectos mais curtos, 1997. (Dissertação de Mestrado; Departamento de Matemática; Universidade de Coimbra.)
(http://www.mat.uc.pt/∼marta/Publicacoes/tese.ps.gz)

21. J.L.E. Santos. O problema do trajecto óptimo multiobjectivo, 1997. (Dissertação de Mestrado; Departamento de Matemática; Universidade de Coimbra.)
(http://www.mat.uc.pt/∼zeluis/INVESTIGACAO/mestrado.ps.gz).

22. D. Shier. Computational experience with an algorithm for finding the k shortest paths in a network. *Journal of Research of the NBS*, 78:139–164, 1974.

23. D. Shier. Interactive methods for determining the k shortest paths in a network. *Networks*, 6:151–159, 1976.

24. D. Shier. On algorithms for finding the k shortest paths in a network. *Networks*, 9:195–214, 1979.

25. J.W. Suurballe. Disjoint paths in a network. *Networks*, 4:125–145, 1974.

26. N. Wirth. *Algorithms + Data Strutures = Programs.* Prentice-Hall, Inc., 1976.

27. J.Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17:712–716, 1971.