AN ALGORITHM FOR RANKING LOOPLESS PATHS

Ernesto de Queirós Vieira Martins
Marta Margarida Braz Pascoal
and
José Luis Esteves dos Santos

{*eqvm,marta,zeluis*}*@mat.uc.pt*
Centro de Informática e Sistemas
Departamento de Matemática
Universidade de Coimbra
PORTUGAL

December 1999

Abstract: In this paper an algorithm for the ranking of loopless paths problem is proposed which is valid for directed and undirected networks. Despite its theoretical computational complexity be an open problem yet, the algorithm appears to perform well in practice as the reported comparative computational experiments allow us to conclude. This conclusion is reinforced with some results obtained with larger networks; more than 500,000 loopless paths were ranked in 10,000 nodes and 100,000 arcs euclidian networks in about 0.35 seconds of CPU execution time when all the arcs are undirected and in about 0.15 seconds for directed euclidian networks, using a server with 128 Mbytes of RAM and a 275 MHz processor running DEC Unix 3.2.

Keywords: path, loopless path, path distance, paths ranking, network.

---

1 Introduction

Let $(\mathcal{N},\mathcal{A})$ denote a given network, where $\mathcal{N} = \{v_1, \ldots, v_n\}$ is a finite set whose elements are called nodes and $\mathcal{A} = \{a_1, \ldots, a_m\}$ is a proper subset of $\mathcal{N} \times \mathcal{N}$ whose elements are called arcs. Each arc a_k is a pair (v_i, v_j) where $v_i \neq v_j$. If (v_i, v_j) is an ordered (unordered) pair for any $(v_i, v_j) \in \mathcal{A}$, then $(\mathcal{N},\mathcal{A})$ is said to be a directed (undirected) network. Let $a_k = (v_i, v_j)$ be a directed arc; then v_i called the tail node of a_k and v_j its head node.

Let s, t be two given nodes of $(\mathcal{N},\mathcal{A})$. A non null path p (or path p) from s to t in $(\mathcal{N},\mathcal{A})$ is a sequence of nodes and arcs of the form $p = \langle v'_1, a'_1, v'_2 \ldots, v'_\ell, a'_\ell, v'_{\ell+1} \rangle$, such that:

- $v'_i \in \mathcal{N}$ for any $i \in \{1, \ldots, \ell + 1\}$;
- $a'_k = (v'_k, v'_{k+1}) \in \mathcal{A}$ for any $k \in \{1, \ldots, \ell\}$;
- $v'_1 = s$ and $v'_{\ell+1} = t$.

To simplify the notation, sometimes $i \in \mathcal{N}$ is used to denote node $v_i \in \mathcal{N}$.

A cycle (or loop) is a sequence p for which all nodes are different except the first one which is also the last (that is, $s = t$).

A path p is said to be loopless *if and only if* all its nodes are different. A null path is a sequence with a single node.

For any arc $a_k = (v_i, v_j) \in \mathcal{A}$, let $c_{a_k} \in \mathbb{R}$ or $c_{ij} \in \mathbb{R}$ – known as the cost (or distance) of arc $a_k = (v_i, v_j)$ – denotes a real number associated with $a_k = (v_i, v_j)$, and let $\mathcal{P}_{ij}$ be the set of paths from v_i to v_j in ($\mathcal{N}$,$\mathcal{A}$), for any $v_i, v_j \in \mathcal{N}$. To simplify, $\mathcal{P}$ will be used to denote $\mathcal{P}_{st}$.

In what follows and with no loss of generality it is assumed that $\mathcal{P}_{si} \neq \emptyset$ and $\mathcal{P}_{it} \neq \emptyset$, for any $i \in \mathcal{N}$. Usually $c(p)$ is called the cost (or distance) of path p.

Let $c : \bigcup_{i,j \in \mathcal{N}} \mathcal{P}_{ij} \longrightarrow \mathbb{R}: p \longrightarrow c(p) = \sum_{(i,j) \in \mathcal{A} \cap p} c_{ij}$ be a real function defined over the set of paths that can be determined in ($\mathcal{N}$,$\mathcal{A}$), that is, a function which assigns a real value to each path from i to j in ($\mathcal{N}$,$\mathcal{A}$), for any $i, j \in \mathcal{N}$.

Let $\mathcal{P}^\star \subseteq \mathcal{P}$ be some subset of $\mathcal{P}$. In the ranking paths problem, for a given positive integer K ($K \leq \mid \mathcal{P}^\star \mid$) it is intended to determine a set $\mathcal{P}_K = \{p_1, \ldots, p_K\} \subseteq \mathcal{P}^\star$, such as:

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

This is a classical problem that can be divided into two classes: the unconstrained problem and the constrained one. In the unconstrained problem $\mathcal{P}^\star = \mathcal{P}$; so, in the constrained ranking paths problem $\mathcal{P}^\star \subset \mathcal{P}$, since all the paths in $\mathcal{P}_K$ have to satisfy some condition; for instance, to be loopless.

Ranking unconstrained paths is considered an easier problem than the ranking loopless paths problem and very efficient algorithms exist to solve it. The interested reader may addresses to [1, 3, 7, 8, 10] or to a complete bibliography due to Eppstein that is online with the following URL address: `http://liinwww.ira.uka.de/bibliography/Theory/k-path.html`.

In a previous paper, [8], the algorithms for ranking shortest paths were divided in two classes: the labeling and the deviation algorithms. While the former are supported by an Optimality Principle generalization, [8, 9], deviation algorithms build a tree of paths containing the K shortest ones. It was also shown that the ranking loopless shortest paths problem does not satisfy, in general, the Optimality Principle generalization, [8]. That is, labeling algorithms can not be used to solve this problem. Nevertheless, all the theory justifying deviation algorithms remains valid when $\mathcal{P}^\star$ is the set of loopless paths from s to t. That is, deviation algorithms can be adapted for ranking loopless paths. As examples of algorithms for ranking loopless paths we point out the classical Yen's algorithm, [6, 11], valid for directed and undirected networks, and an algorithm due to Katoh, Ibaraki and Mine, [5], that is valid only for undirected networks. For the adaptation of Yen's algorithm for solving the unconstrained problem the reader is addressed to [8].

A very important operation in deviation algorithms is the *path concatenation*. So, for any $i, j, \ell \in \mathcal{N}$, let $p \in \mathcal{P}_{i\ell}$ and $q \in \mathcal{P}_{\ell j}$ be two paths from i to ℓ and from ℓ to j, respectively. Since the terminal node of p is also the initial node of q, the p and q *concatenation*, denoted by $p \diamond q$, can be defined. Moreover, $p \diamond q \in \mathcal{P}_{ij}$, that is, $p \diamond q$ is the path from i to j which coincides with p from i to ℓ and with q from ℓ to j. Obviously if p (or q) is a null path then $p \diamond q = q$ (or $p \diamond q = p$).

Sometimes it will also be referred the concatenation of a path $p \in \mathcal{P}_{ij}$ with an arc (j,k) to mean the paths concatenation $p \diamond \langle j,(j,k),k\rangle$.

It results trivial to conclude that the loopless paths concatenation is not necessarily a loopless path. This is the main difference between deviation algorithms for ranking paths and deviation algorithms for ranking loopless paths.

The characteristic of all deviation algorithms is a set X of candidate paths to the next shortest path. That is, in the $k^{\underline{th}}$ step of a deviation algorithm for the unconstrained problem, the $k^{\underline{th}}$ shortest path is the shortest path of the set X. In general, this set X is being actualized by joining new paths which result from the concatenation $p_{si} \diamond p_{it}$, where p_{si} is the subpath of p_k, from s to a well determined node i of p_k, and p_{it} is a constrained shortest path from i to t since p_{it} has to be determined in such a way that $p_{si} \diamond p_{it} \notin \mathcal{P}_k = \{p_1 \ldots, p_k\}$. That is, for well determined nodes i of p_k, a constrained shortest path problem from i to t has to be solved. When paths are constrained to be loopless, it appears that $p_{si} \diamond p_{it}$ has to be a loopless path too. This is the case of the algorithm of Yen and the algorithm of Katoh *et al.* in which X is kept free of paths with loops, that is, $p_{si} \diamond p_{it}$ is always a loopless path. This is achieved, by the removal from the network of all the nodes of p_{si} but i, all the arcs whose tail or/and head node has been removed and some well determined arcs $(i,j) \in \mathcal{A}$; more precisely, the set of arcs $(i,j) \in \mathcal{A}$ for which $p = p_{si} \diamond \langle i,(i,j),j\rangle \diamond p^{\star}_{jt} \in \mathcal{P}_k$, where $\langle i,(i,j),j\rangle \diamond p^{\star}_{jt}$ is a shortest path from i to t in the resulting network. Obviously, the given initial network has to be reconstructed after the determination of such a constrained shortest path.

In this paper an algorithm is proposed for the loopless paths problem that can be viewed as an adaptation of MPS algorithm for the unconstrained problem, [8]. This adaptation solves the ranking loopless shortest paths problem, but paths with loops may be included in X, since with the loopless paths concatenation the determination of a loopless path is not assured. So, assuming that $\mathcal{P}_{k-1}$ was already determined, p, the shortest path of X, is the $k^{\underline{th}}$ shortest loopless path *if and only if* p is a loopless path. This disadvantage appears to be compensated since the constrained shortest path problem is very easily solved in MPS algorithm.

Computational experiments with randomly generated undirected networks, comparing this new algorithm with the classical Yen's algorithm and the algorithm of Katoh *et al.* are reported, as well as computational experiments showing some performances of the algorithm in the resolution of larger problems.

Throughout it is assumed that $\sum_{(i,j)\in C} c_{ij} \geq 0$ (or $\sum_{C} c_{ij} \geq 0$ for reasons of simplicity) for any cycle C if $(\mathcal{N},\mathcal{A})$ is a directed network. As a consequence, if $(\mathcal{N},\mathcal{A})$ is an undirected network, it is assumed that $c_{ij} \geq 0$ holds for any arc $(i,j) \in \mathcal{A}$.

2 Mathematical Support

The algorithm uses the change of costs firstly used by Eppstein in its algorithm for the unconstrained problem. With this change of costs the number of arithmetic operations that have to be executed is significantly decreased.

2.1 A change of costs

Let $\mathcal{T}$ be a spanning tree of $(\mathcal{N},\mathcal{A})$ rooted at t, that is, a tree whose set of nodes is $\mathcal{N}$, the set of arcs is a subset of $\mathcal{A}$ and there is a single (loopless) path from i to t, for any $i \in \mathcal{N}$. Let $\pi_i(\mathcal{T})$

be the cost of the path from any node $i \in \mathcal{N}$ to t which is determined in $\mathcal{T}$; so, $\pi_t(\mathcal{T}) = 0$. Let $\mathcal{T}_t$ be the shortest tree rooted at t, that is, a spanning tree of $(\mathcal{N},\mathcal{A})$ rooted at t, for which its path from i to t is a shortest path in $\mathcal{P}_{it}$, for any $i \in \mathcal{N}$. In what follows, π_i will be used for $\pi_i(\mathcal{T}_t)$, for any $i \in \mathcal{N}$.

Lemma 2.1 *Let $\pi_i(\mathcal{T})$ be the cost of the loopless path from i to t that is determined in $\mathcal{T}$, a spanning tree of $(\mathcal{N},\mathcal{A})$ rooted at t, and let $\bar{c}_{ij}(\mathcal{T}) = \pi_j(\mathcal{T}) - \pi_i(\mathcal{T}) + c_{ij}$, for any $(i,j) \in \mathcal{A}$. Moreover let $\bar{c}_{\mathcal{T}}(p) = \sum_p \bar{c}_{ij}(\mathcal{T})$. Then, $\bar{c}_{\mathcal{T}}(p) = \pi_t(\mathcal{T}) - \pi_s(\mathcal{T}) + c(p)$ holds for any $p \in \mathcal{P}$.*

Proof: Let $\mathcal{T}$ be a spanning tree of $(\mathcal{N},\mathcal{A})$ rooted at t. Since, $\bar{c}_{\mathcal{T}}(p) = \sum_p \bar{c}_{ij}(\mathcal{T})$, then $\bar{c}_{\mathcal{T}}(p) = \sum_p (\pi_j(\mathcal{T}) - \pi_i(\mathcal{T}) + c_{ij})$. For any path p, only $\pi_s(\mathcal{T})$ and $\pi_t(\mathcal{T})$ belong exactly once to the equation $\bar{c}_{\mathcal{T}}(p) = \sum_p (\pi_j(\mathcal{T}) - \pi_i(\mathcal{T}) + c_{ij})$. For all the intermediate nodes j of p, if any, $\pi_j(\mathcal{T})$ appears exactly twice – when the arc (i,j) and the arc (j,ℓ), that follows (i,j) in path p, are considered. So, 1 and -1 are the coefficients of $\pi_j(\mathcal{T})$ in the summation and the lemma follows. ∎

When $\mathcal{T}$ is the shortest tree rooted at t, it will be used $\bar{c}_{ij}$ and $\bar{c}(p)$ for $\bar{c}_{ij}(\mathcal{T})$ and $\bar{c}_{\mathcal{T}}(p)$, respectively. Note that the problem of the shortest tree rooted at t is assumed to be known; among the hundreds of references, the reader may address [4] for details.

The following theorems are the theoretical support of MPS algorithm. Despite being well known from Linear Programming, they will be stated and proved, anyway.

Theorem 2.1 *Let $\mathcal{T}_t$ be a shortest tree rooted at t and let p_{it} be the shortest path from i to t that is determined in $\mathcal{T}_t$, for any $i \in \mathcal{N}$. Then:*

1. *$\bar{c}_{ij} \geq 0$, for any arc $(i,j) \in \mathcal{A}$;*
2. *$\bar{c}_{ij} = 0$, for any arc $(i,j) \in \mathcal{A} \cap \mathcal{T}_t$;*
3. *$\bar{c}(p_{it}) = 0$, for any node $i \in \mathcal{N}$.*

Proof:

1. Let $(i,j) \in \mathcal{A}$ be an arc of $(\mathcal{N},\mathcal{A})$. So, let us assume that $\bar{c}_{ij} = \pi_j - \pi_i + c_{ij} < 0$; that is, $c_{ij} + \pi_j < \pi_i$. Since, by definition, π_j is the cost of all shortest paths from j to t, π_j is the cost of p_{jt} – the path from j to t that it is determined in $\mathcal{T}_t$. Moreover, $q = \langle i,(i,j),j\rangle \diamond p_{jt} \in \mathcal{P}_{it}$ is a path from i to t and $c(q) = c_{ij} + \pi_j < \pi_i$, which contradicts the definition of π_i – the cost of all shortest paths from i to t.

2. From the first part of the theorem, $\bar{c}_{ij} \geq 0$ holds for any $(i,j) \in \mathcal{A}$. That is, $c_{ij} + \pi_j \geq \pi_i$, for any $(i,j) \in \mathcal{A}$. As $(i,j) \in \mathcal{A} \cap \mathcal{T}_t$, then $q = \langle i,(i,j),j\rangle \diamond p_{jt} \in \mathcal{T}_t$ and q is also a shortest path from i to t; that is, $c(q) \leq c(p_{it})$ or $c_{ij} + \pi_j \leq \pi_i$. So, $c_{ij} + \pi_j = \pi_i$ and $\bar{c}_{ij} = 0$.

3. As p_{it} is the path from i to t that is determined in $\mathcal{T}_t$, the proof results immediate from the second part of the theorem. ∎

The following theorem is very important for supporting the change of costs that was defined in lemma 2.1.

Theorem 2.2 *Let $\mathcal{T}_t$ be a shortest tree of $(\mathcal{N},\mathcal{A})$ rooted at t and let $p, q \in \mathcal{P}$ be two (loopless) paths from s to t in $(\mathcal{N},\mathcal{A})$. Then, $c(p) \leq c(q)$ if and only if $\bar{c}(p) \leq \bar{c}(q)$.*
Proof: Let $p, q \in \mathcal{P}$ be two (loopless) paths defined from s to t in $(\mathcal{N},\mathcal{A})$. From lemma 2.1, $\bar{c}(p^\star) = \pi_t - \pi_s + c(p^\star)$, for any $p^\star \in \mathcal{P}$. So, $\bar{c}(p) \leq \bar{c}(q)$ if and only if $\pi_t - \pi_s + c(p) \leq \pi_t - \pi_s + c(q)$, that is, if and only if, $c(p) \leq c(q)$. ∎

Note that theorem 2.2 could be generalized for some spanning tree $\mathcal{T}_\theta$ of $(\mathcal{N},\mathcal{A})$ rooted at θ, for any node $\theta \in \mathcal{N}$. Note also that, from theorem 2.2, c_{ij} can be replaced by $\bar{c}_{ij}$, for any $(i,j) \in \mathcal{A}$, without changing the optimal solution of the ranking of shortest (loopless) paths problem. Moreover, and as it will be shown later on, this changing of costs allows the reduction of the number of arithmetic operations that have to be executed by the algorithm.

Notice that, according to part 3 of theorem 2.1, $\bar{c}(p^i_{\mathcal{T}_t}) = 0$ holds for any $i \in \mathcal{N}$, where $p^i_{\mathcal{T}_t}$ denotes the shortest (loopless) path from i to t that is determined in $\mathcal{T}_t$.

2.2 The set of candidate paths

In this subsection the role of X, the set of candidate paths in the algorithm, as well as X is being actualized, will be explained.

As it was previously referred, in the $k^{\underline{th}}$ step of the algorithm for the unconstrained ranking problem, p_k is the shortest path of X. As a consequence, X is initialized with $p_1 = p^s_{\mathcal{T}_t}$ – the shortest path from s to t determined in $\mathcal{T}_t$. In the ranking of the K shortest loopless paths problem, since paths with loops are allowed to belong to X, its shortest loopless path is the next loopless path to be computed in the ranking of loopless paths problem. Furthermore, assuming the nonexistence of negative cycles in $(\mathcal{N},\mathcal{A})$, that is, cycles $\mathcal{C}$ such that $\sum_{\mathcal{C}} c_{ij} < 0$, it may be assumed too, and with no loss of generality, that the shortest path p_1 is also a loopless path, even when non loopless shortest paths may exist in $(\mathcal{N},\mathcal{A})$. That is, X is also initialized with p_1 – a shortest loopless path from s to t in $(\mathcal{N},\mathcal{A})$.

Actualizing X implies joining to X some well determined paths $q \in \mathcal{P}$. It is important to emphasize the existence of a well determined node associated with each path $q \in \mathcal{P} \cap X$. It will be called *the deviation node of q* and it is recursively determined. By assumption, the initial node s is the deviation node of p_1 — the (loopless) shortest path that initializes set X. Let $q \in \mathcal{P} \cap X$ be a path from s to t that belongs to X. One arc is also associated with each node $i \in \mathcal{N}$ of each path q, which will be called the *last arc of q at node $i \in \mathcal{N}$*; actually the last arc of q at $i \in \mathcal{N}$ is the arc of q whose tail node is $i \in \mathcal{N}$.

In general and similarly to MPS algorithm for the unconstrained ranking paths problem, let $p \in \mathcal{P} \cap X$ be the shortest path from s to t in the set of paths belonging to X; that is, $c(p) \leq c(q)$ holds for any $q \in X$. Then p will be removed from X and it will be the next shortest loopless path, *if and only if*, p is loopless. In any case, in general, some paths are generated from p and such generated paths will be joined to X.

Let d_p be the deviation node of $p \in X$ and let p_{xy} denote the subpath of p from x to y. For each node i of $p_{d_p t}$ such that $i \neq t$, a constrained shortest path problem has to be solved since a

(loopless) path $q_{it} \in \mathcal{P}_{it}$ has to be determined in such a way that q_{it} is a shortest (loopless) path from i to t and $p_{si} \diamond q_{it}$ does not belong neither did ever belong to X. Obviously, it may happen that no solution exists to this constrained shortest path problem. However, as it will be shown, it is very easy to conclude that no such path exists or, otherwise, it very easy to determine it. Obviously, when a path $q_{it} \in \mathcal{P}_{it}$ satisfying the imposed conditions exists, $p_{si} \diamond q_{it}$ will be joined to X (even if $p_{si} \diamond q_{it}$ is not a loopless path); moreover, $i \in \mathcal{N}$ is its deviation node.

In the next subsection it will be shown how an appropriated data structure, when used for defining $(\mathcal{N},\mathcal{A})$, allows the immediate determination of $q_{it} \in \mathcal{P}_{it}$ in such a way that $p_{si} \diamond q_{it} \in \mathcal{P}$ is a candidate path or to conclude that no such a constrained shortest path exist.

2.3 The sorted forward star form

Let $(\mathcal{N},\mathcal{A})$ be a given network and let us represent $\mathcal{N} = \{v_1, \ldots, v_n\}$, its set of nodes, as $\{1, \ldots, n\}$. Let us represent $\mathcal{A}$, the set of arcs of $(\mathcal{N},\mathcal{A})$, in such a way that $\mathcal{A} = \mathcal{A}(1) \cup \ldots \cup \mathcal{A}(n)$, where $\mathcal{A}(k)$ is the set of arcs whose tail node is k, for any $k \in \mathcal{N}$. So, $\mathcal{A}(i) \cap \mathcal{A}(j) = \emptyset$, for any $i, j \in \mathcal{N}$ such that $i \neq j$. Moreover, let $a'_k, a'_\ell \in \mathcal{A}$ such that $a'_k \in \mathcal{A}(\xi)$ and $a'_\ell \in \mathcal{A}(\theta)$. Let us assume that $\xi \neq \theta$; then $a'_k < a'_\ell$ *if and only if* $\xi < \theta$. Let us assume now that $\xi = \theta$; then $a'_k < a'_\ell$ *if* $\bar{c}(a'_k) \leq \bar{c}(a'_\ell)$. That is, the set $\mathcal{A}$ is sorted in such a way that, for any two arcs $(k, j), (i, \ell) \in \mathcal{A}$, $(k, j) < (i, \ell)$ *if* $k < i$ or ($k = i$ and $\bar{c}_{kj} \leq \bar{c}_{i\ell}$). This is the so called *sorted forward star form*, [2]. It results that when $\mathcal{A} = \{a_1, \ldots, a_m\}$ is arranged in the sorted forward star form, then $\bar{c}(a_k) > \bar{c}(a_{k+1})$ *only if* the tail node of a_k is less than the tail node of a_{k+1}. Moreover, *if* the tail nodes of a_k and a_{k+1} are the same *then* $\bar{c}(a_k) \leq \bar{c}(a_{k+1})$. This property of the sorted forward star form allows the immediate determination of a path $p_{si} \diamond q_{it}$ satisfying the condition stated in subsection 2.2. In fact, for any path p just picked out in X, each one of its nodes i ($i \neq t$) following d_p has to be analized considering $q_{it} = \langle i, (i, j), j\rangle \diamond p^j_{\mathcal{T}_t}$, where, recall, $p^j_{\mathcal{T}_t}$ is the shortest path from j to t in $\mathcal{T}_t$. That is, $\bar{c}(p^j_{\mathcal{T}_t}) = 0$ for any node $j \in \mathcal{N}$. So, the constrained shortest path problem that has to be solved is just the determination of the arc (i, j) for which $\bar{c}_{ij}$ is minimized over the set $\mathcal{A}(i)$ – of the arcs emerging from $i \in \mathcal{N}$ – and such that $p_{si} \diamond \langle i, (i, j), j\rangle \diamond p^j_{\mathcal{T}_t}$ did never belong to X. As $(\mathcal{N},\mathcal{A})$ is described in the sorted forward star form, it is immediate to conclude that the minimizing arc (i, j), if such an arc exists, is the one which follows the last arc of p at $i \in \mathcal{N}$ and its tail node is also $i \in \mathcal{N}$. Recall that paths with loops are allowed to belong to X; however, to assure the finiteness of the algorithm and to decrease the number of non feasible paths in X, (i, j) will be chosen as the one which follows the last arc of p at $i \in \mathcal{N}$ but in such a way that $p_{si} \diamond \langle i, (i, j), j\rangle$ is loopless. Note that it is possible that such an arc does not exist and, in this case, no path is joined to X. It is also immediate to conclude how the number of arithmetic operations is decreased; in fact, since $\bar{c}(p^j_{\mathcal{T}_t}) = 0$ for any $j \in \mathcal{N}$, then $\bar{c}(p_{si} \diamond \langle i, (i, j), j\rangle \diamond p^j_{\mathcal{T}_t}) = \bar{c}(p_{si}) + \bar{c}_{ij}$.

It is important to point out that the first arc of $\mathcal{A}(i)$ has to be an arc of $\mathcal{T}_t$, for any $i \in \mathcal{N}$.

3 The Algorithm

The algorithm is now described. We must point out that it is adapted from MPS algorithm for the unconstrained ranking paths problem, [8].

Theorem 3.1 validates the algorithm.

Begin {algorithm}
Determine $\mathcal{T}_t$, the tree of shortest paths from i to t, for any $i \in \mathcal{N}$;
$p_1 \longleftarrow$ path from s to t in $\mathcal{T}_t$;
$\bar{c}_{ij} \longleftarrow \pi_j - \pi_i + c_{ij}$, for any $(i,j) \in \mathcal{A}$;
$\mathcal{A} \longleftarrow \{a_1, \ldots, a_m\}$ where, for any $\ell \in \{1, \ldots, m-1\}$, $\bar{c}_{a_\ell} \leq \bar{c}_{a_{\ell+1}}$ if $\text{tail}(a_\ell) = \text{tail}(a_{\ell+1})$;
$X \longleftarrow \{p_1\}$; $k \longleftarrow 0$;
While ($X \neq \emptyset$ and $k \leq K$) **do**
begin
$p \longleftarrow$ path of X such that $\bar{c}(p) \leq \bar{c}(q)$ for any $q \in X$;
$X \longleftarrow X - \{p\}$;
if (p has no cycles) **then**
begin
$k \longleftarrow k + 1$;
$p_k \longleftarrow p$
end;
$v_i \longleftarrow$ deviation node of p;
Repeat
$p_{sv_i} \longleftarrow$ the subpath of p, from s to v_i;
$a_\ell \longleftarrow p$ arc whose tail node is v_i; /$\star$ last arc of p at v_i $\star$/
while (v_i is the tail node of a_{h+1}) **and** (a_{h+1} forms a cycle with p_{v_i}) **do** $h \longleftarrow h + 1$;
if (v_i is the tail node of a_{h+1}) **then**
begin
$v_j \longleftarrow$ head node of a_{h+1};
$p_{\mathcal{T}_t}^{v_j} \longleftarrow$ shortest path from v_j to t in $\mathcal{T}_t$;
$X \longleftarrow X \cup (p_{sv_i} \diamond \langle v_i, a_{h+1}, v_j \rangle \diamond p_{\mathcal{T}_t}^{v_j})$
end;
$v_i \longleftarrow$ next node in p
until (p_{sv_i} has a cycle) **or** ($v_i = t$)
end
End. {algorithm}

Theorem 3.1 *The set $\mathcal{P}_K = \{p_1, \ldots, p_K\}$ of the K shortest loopless paths is determined with the algorithm.*

Proof: $\mathcal{T}_t$, the tree of the shortest (loopless) path rooted at t is determined with one of the existing algorithms for the problem, [2, 4]. Consequently, the shortest path p_1 is determined.

Let us assume that $\mathcal{P}_{k-1} = \{p_1, \ldots, p_{k-1}\}$ is also determined and let us prove that p_k is determined too. For every $i \in \{1, \ldots, k-1\}$ it is immediate that p_k coincides with p_i from the initial node s until a node v_k^i which can be the whole node s. That is, $p_{sv_k^i}$ is a subpath of p_k and a subpath of p_i for any $i \in \{1, \ldots, k-1\}$. Among these $k-1$ subpaths of p_k, let $p_{sv_k^\theta}$ be the one with greater number of arcs. It is immediate to verify that v_k^θ is the deviation node of p_k. It is also immediate that p_θ ($\theta < k$) belonged to X and its node v_k^θ was analized by the algorithm. Let us assume that p_k was not the path originated by p_θ at node v_k^θ. That is, at this step of the algorithm a path p was joined to X such that:

- v_k^θ is the deviation node of p;
- $p = p_{sv_k^\theta} \diamond \langle v_k^\theta, (v_k^\theta, j), j\rangle \diamond p^j_{\mathcal{T}_t}$;
- $p_{sv_k^\theta} \diamond \langle v_k^\theta, (v_k^\theta, j), j\rangle$ is loopless;
- $c(p) \leq c(p_k)$;
- p is not loopless.

That is, p is a nonloopless path that belonged to X, such that $c(p) \leq c(p_k)$. Since p_k, by assumption, was not determined, it can be concluded that p was chosen in X (otherwise $k > K$) and all its nodes i ($i \neq t$) of its subpath $p_{v_k^i t}$ were analized. The theorem follows because the number of arcs of $\mathcal{A}(v_k^\theta)$ is finite as well as the number of nodes in any loopless path. Recall that in the last case no more arcs are concatenated with a loopless path in order that a loopless path is also obtained. ∎

It must be pointed out that, from the proof of theorem 3.1, it is obvious to conclude about the finiteness of the algorithm. Notice that this fact is assured because the minimizing arc (i, j) is that one whose concatenation with a loopless path results in a loopless path too.

From the algorithm is also immediate to conclude that p_k is determined before p_{k+1}, for any $k \geq 1$.

4 Computational Experiments

A code for the algorithm described in this paper (**MPS** code) was compared with a code for Yen's algorithm (**YEN** code) and a code for the algorithm of Katoh, Ibaraki and Mine (**KIM** code). All the codes were written in C and the computational experiments were carried out on a DEC AXP 7610 server with a 275 MHz processor running DEC Unix 3.2. The server has 128 Mbytes of RAM and 200.9 SPECint92.

In the comparative computational experiments we used two different types of randomly generated undirected networks.

For each value of *down*, ten problems were generated, $K = 1000$ loopless paths were ranked by each one of the three codes and the average execution times computed for each loopless path. The obtained results were very similar; so we only report (figure 3) the results for 100×25 grid networks.

The superiority of MPS code over both KIM and YEN codes seems to be obvious (for $K = 1000$, MPS code runs from 500 to 1000 times faster than the best of the other two). Also the linearity of the results for all the codes is an immediate conclusion – the average **MPS** code times are always constant.

We must point out that the needed RAM space is not the same for all the codes; in fact, according to this parameter, KIM code is the best followed by MPS code. The reason why a peculiar shape for the average running times was obtained with YEN code, results from the fact that it could not even solve all the ten 100×25 grid problems until $K = 1000$.

For $K \leq 1000$, the MPS code running times are constant for each solved problem. This fact suggested the resolution of larger problems in order to check the possibilities of MPS code. Our

intent was to determine not only the number of loopless paths that could be ranked with MPS code, but also the needed CPU running time and RAM space. In these experiments we used (directed and undirected) euclidean, grid and complete networks. The results for five problems randomly generated for each instance, are presented in table 1.

We must point out that MPS code (as well as YEN code) considers all the networks as being directed since each undirected arc is equivalent to a pair of arcs with equal distance and opposite directions. According to table 1 the results concerning the CPU execution time, being directed or undirected seems to be unimportant only for complete networks. Taking into account the number K of ranked loopless paths, problems in directed networks seems to be easier although grid networks presents no peculiar results.

An obvious conclusion follows from the network topology; in fact, grid network problems are harder to solve, since paths generated in this problems have a lot of arcs which implies more memory space and worse execution times.

Density also appears to be important when we are concerned with the execution time. For example, the average execution times for complete networks are worse than those obtained for euclidean networks, for example. However, we can not forget that density is an important parameter in the determination of the shortest tree as well as when the change of costs is executed and when the set of arcs is sorted (which is confirmed by the results in column **R.T.** of table 1).

5 Conclusion

In this paper a new algorithm for ranking loopless paths was proposed. In order to determine its theoretical complexity, an upper bound for the number of paths that needs to be generated for ranking K loopless paths has to be determined which is an open problem yet, despite the number of candidate paths being finite. However, the computational experiments are a reasonable measure of the average behavior of an algorithm and they show the efficiency of the proposed one as well as its superiority over two classical algorithms, which, to the best of our knowledge, are being considered the most efficient.

Acknowledgments:
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").

	$K = 1$		$K = 100000$		**last** K				**R.T.**	
	seconds				K		seconds		seconds	
	und.	dir.	und.	dir.	und.	dir.	und.	dir.	und.	dir.
complete 1000 nodes	1.019	1.098	2.035	2.127	609099	742717	2.042	2.135	0.008	0.009
	0.976	0.859	2.006	1.880	546775	837756	2.014	1.887	0.010	0.008
	1.000	0.991	2.034	2.017	650736	828785	2.042	2.024	0.010	0.008
	0.965	1.077	1.899	1.969	521609	739092	1.907	1.976	0.010	0.008
	1.016	0.941	2.042	1.976	578996	808513	2.051	1.985	0.011	0.010
average	1.008	0.978	2.003	1.994	581443.	783329.	2.011	2.001	0.010	0.009
euclidean 10000 nodes 100000 arcs	0.166	0.068	0.324	0.130	599193	621475	0.334	0.144	0.012	0.016
	0.149	0.066	0.299	0.130	574896	646515	0.309	0.145	0.012	0.016
	0.184	0.073	0.335	0.133	441301	603677	0.346	0.148	0.014	0.016
	0.173	0.071	0.322	0.129	526221	630502	0.334	0.146	0.014	0.017
	0.166	0.064	0.316	0.125	530606	610426	0.327	0.138	0.013	0.014
average	0.168	0.068	0.319	0.129	534443.	622519.	0.330	0.144	0.013	0.016
1000×25 grid	2.034	2.005	d.n.r.	d.n.r.	30275	10743	2.154	2.216	0.075	0.164
	1.962	2.170	d.n.r.	d.n.r.	14441	13591	2.055	2.283	0.081	0.065
	2.122	2.183	d.n.r.	d.n.r.	16636	49314	2.238	2.316	0.068	0.084
	1.736	1.923	d.n.r.	d.n.r.	23579	44671	1.873	2.046	0.090	0.075
	1.878	2.225	d.n.r.	d.n.r.	31996	17557	2.001	2.329	0.077	0.055
average	1.946	2.101	——	——	23385.	27175.	2.064	2.238	0.078	0.089
150×150 grid	0.156	0.168	0.211	0.217	107044	210464	0.213	0.225	0.016	0.015
	0.154	0.152	d.n.r.	d.n.r.	92535	50832	0.216	0.204	0.021	0.010
	0.141	0.176	d.n.r.	d.n.r.	97133	98567	0.197	0.235	0.014	0.015
	0.210	0.167	d.n.r.	d.n.r.	85051	80173	0.266	0.220	0.014	0.011
	0.173	0.179	d.n.r.	d.n.r.	87537	68167	0.225	0.231	0.010	0.009
average	0.167	0.168	——	——	93862.	101641.	0.223	0.223	0.015	0.012

Complete network problems used about **125 Mb** of RAM.
Euclidean and grid network problems used about **105 Mb** of RAM.

und. – undirected; **dir.** – directed; **d.n.r.** – did not run until the end;
R.T. – Total time – (shortest tree time + change of costs time + arcs sort time).

Table 1: **MPS code results.**

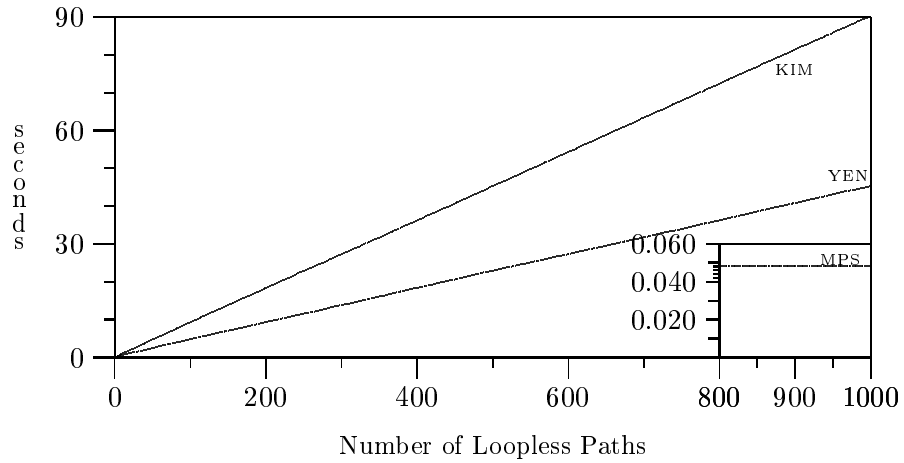

Figure 1: **10000 nodes and 20000 arcs euclidean networks – zooming for** $K \geq 800$**.**

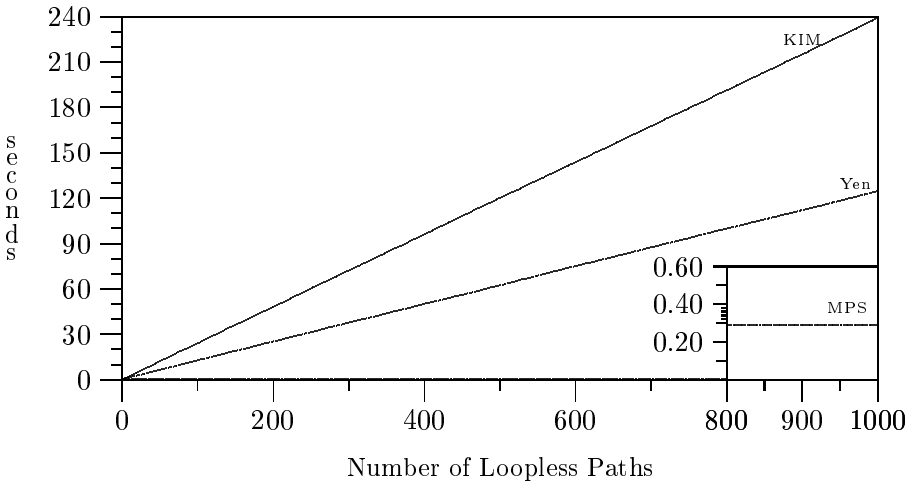

Figure 2: **10000 nodes and 100000 arcs euclidean networks – zooming for** $K \geq 800$**.**

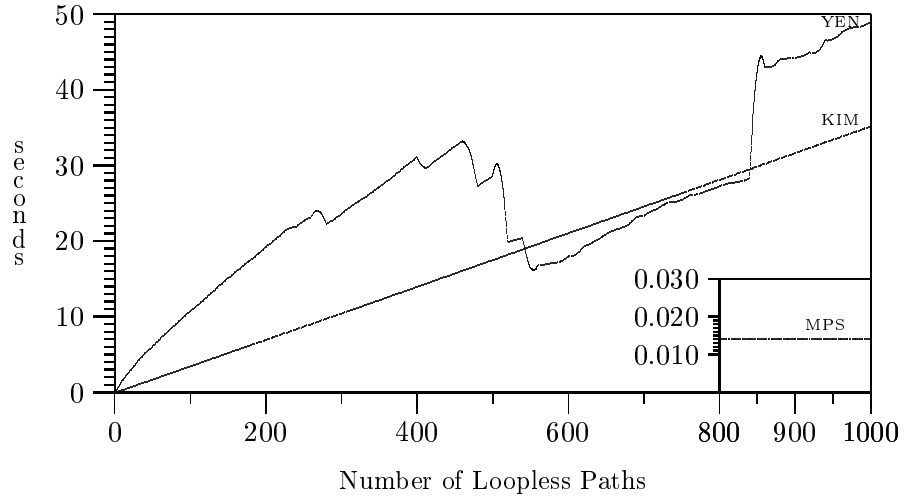

Figure 3: 100×25 **grid networks – zooming for** $K \geq 800$**.**

References

[1] 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, (1994), 188–191.

[2] 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, (1979), 215–348.

[3] D. Eppstein, Finding the k Shortest Paths, 35th IEEE Symp. Foundations of Comp. Sci., Santa Fé (1994), 154–165.

[4] G. Gallo and S. Pallotino, Shortest path methods: a unifying approach, Mathematical Programming Study 26, (1986), 38–64.

[5] N. Katoh, T. Ibaraki and H. Mine, An efficient algorithm for K shortest simple paths, Networks 12, (1982), 411–427.

[6] E.L. Lawler, A Procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem, Management Science 18, (1972), 401–405.

[7] E.Q.V. Martins and J.L.E. Santos, A New Shortest Paths Ranking Algorithm, to appear in *Investigação Operacional*.
(`http://www.mat.uc.pt/~eqvm/cientificos/investigacao/Artigos/K.ps.g`)

[8] E.Q.V. Martins, M.M.B. Pascoal and J.L.E. Santos, Deviation algorithms for ranking shortest paths, Journal of Foundations of Computer Science 10 (3), (1999), 247–263.
(`http://www.mat.uc.pt/~eqvm/cientificos/investigacao/Artigos/deviation.ps.gz`)

[9] E.Q.V. Martins, M.M.B. Pascoal and J.L.E. Santos, Labeling algorithms for ranking shortest paths, submitted.

[10] D. Shier, On algorithms for finding the K shortest paths in a network, Networks 9, (1979), 195–214.

[11] J.Y. Yen, Finding the K shortest loopless paths in a network, Management Science 17, (1971), 712–716.