AN ALGORITHM FOR RANKING OPTIMAL PATHS[1]

Ernesto de Queirós Vieira Martins[(2)]
and
Marta Margarida Braz Pascoal[(1,2)]

{*eqvm, marta*}*@mat.uc.pt*
[(1)] Centro de Informática e Sistemas
[(2)] Departamento de Matemática
Universidade de Coimbra
Apartado 3008
3001-454 Coimbra
PORTUGAL
Phone: +351 239 791150, Fax: +351 239 832568

October 2000

Abstract: Assuming the existence of an algorithm for computing the optimal loopless path between two given nodes in a network when considering a general cost function, in this paper a deviation algorithm for enumerating K optimal loopless paths is proposed. It is also presented an adaptation of this algorithm, in order to determine the K optimal paths, possibly containing cycles.

Finally it is shown that the complexity order of this algorithm, when considering a worst-case analysis, is $\mathcal{O}(Kn\,c(n))$, where $c(n)$ stands for the number of performed operations to determine the optimal (loopless) path between a given pair of nodes in a network with n nodes.

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

---

1 Introduction

The enumeration of the K shortest paths is a well-known optimization problem whose purpose is to rank K paths between a given pair of nodes in a network by non-decreasing order of their costs, where the cost value of a path is the summation of all path's arcs costs. When the aim is to rank paths without repeated nodes the problem is known as the ranking of the K shortest loopless paths. Several articles have been published on this subject and the interested reader may consult a bibliography available on-line at `http://liinwww.ira.uka.de/bibliography/Theory/k-path.html`.

Other ranking paths problems can be defined when considering different cost functions, either for determining paths or loopless paths, such as in [3], where Chen proposes an algorithm for ranking quickest loopless paths. Related problems were studied by Lawler, [6], and Murty, [11]. The first one presents an algorithm for ranking the *best* solutions of a discrete optimization problem when any cost function is considered, which intends to be a generalization of the ranking assignments algorithm proposed by Murty.

In this paper an algorithm for ranking paths and loopless paths by non-decreasing order of their costs is presented, valid for any given cost function, as long as an algorithm to compute the optimal path exists.

In sections 2 and 3 some notation and definitions are introduced and the K optimal paths problem is presented. In section 4 a tree structure is defined, which is the support of an algorithm for ranking optimal

[1]The research of Marta Pascoal was developed within CISUC, which is partially supported by the portuguese Ministry of Science and Technology (MCT), under PRAXIS XXI Project of JNICT.

loopless paths, proposed in section 5. In the following section this algorithm is adapted for determining paths possibly with cycles and finally the computational complexity of both algorithms is briefly studied.

2 Notation and definitions

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 $\mathcal{A} = \{a_1, \ldots, a_m\} \subset \mathcal{N} \times \mathcal{N}$ is also a finite set, whose elements are called arcs; each arc a_k can be denoted by (v_i, v_j), with $v_i \neq v_j$. When (v_i, v_j) is an ordered (unordered) pair for any $(v_i, v_j) \in \mathcal{A}$, $(\mathcal{N}, \mathcal{A})$ is said to be a directed (undirected) network. In what follows, and with no loss of generality, $(\mathcal{N}, \mathcal{A})$ will be assumed to be a directed network.

Let i and j be two nodes of $(\mathcal{N}, \mathcal{A})$. A non null path (or simply path) p from i to j in $(\mathcal{N}, \mathcal{A})$ is an alternating sequence of nodes and arcs of the form $p = \langle v'_1 = i, a'_1, v'_2, \ldots, a'_{\ell-1}, v'_\ell = j \rangle$, such that:

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

Nodes i and j are called the initial and terminal nodes of path p, respectively. To simplify the notation, paths can be represented only by their nodes. A null path is formed only by a single node.

A cycle or loop in $(\mathcal{N}, \mathcal{A})$ is a path from one node to itself ($i = j$) where all nodes, except i and j, are different. Therefore a path is said to be loopless when it does not have repeated nodes.

In what follows $\mathcal{P}_{ij}$ will denote the set of paths in $(\mathcal{N}, \mathcal{A})$ from node i to node j.

Given two nodes x and y of some path $p \in \mathcal{P}_{ij}$, $q \in \mathcal{P}_{xy}$ is defined as a subpath of p if it coincides with p from x until y. Such a path will be denoted by $\mathrm{sub}_p(x, y)$. The concatenation of two paths, $p \in \mathcal{P}_{ij}$ and $q \in \mathcal{P}_{j\ell}$, is denoted by $p \diamond q$ and it is the path from i to ℓ formed by path p followed by q.

Let s and t be two different nodes of $(\mathcal{N}, \mathcal{A})$ named, respectively, the initial and the terminal nodes in the network, and let $\mathcal{P}$ denote the set $\mathcal{P}_{st}$.

Let c be a real function defined over the set of all paths in $(\mathcal{N}, \mathcal{A})$, that is, $c : \bigcup_{i,j \in \mathcal{N}} \mathcal{P}_{ij} \longrightarrow \mathrm{I\!R}$. Function c is usually known as path cost function.

In Figure 1 a network $(\mathcal{N}, \mathcal{A})$ with initial node $s = 1$ and terminal node $t = 6$ is represented. With each arc $(i, j) \in \mathcal{A}$ two positive values are associated, denoted by c_{ij} and u_{ij}. In what follows these two values will be called the cost and the capacity of (i, j), respectively.

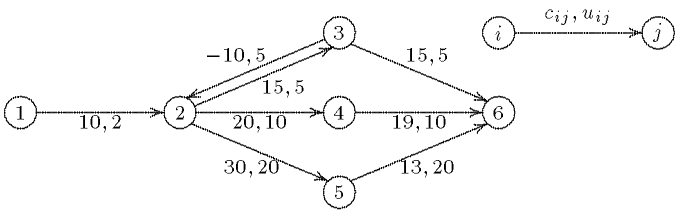

Figure 1: Network $(\mathcal{N}, \mathcal{A})$

Possible cost functions in $(\mathcal{N}, \mathcal{A})$ can be defined by the following expressions, for any path p in the network: $c_1(p) = \sum_{(i,j) \in p} c_{ij}$; $c_2(p) = \min_{(i,j) \in p} \{u_{ij}\}$; $c_3(p) = \dfrac{\sum_{(i,j) \in p} c_{ij}}{\min_{(i,j) \in p} \{u_{ij}\}}$. Table 1 shows four paths from 1 to 6 in $(\mathcal{N}, \mathcal{A})$, one of them non loopless, $\langle 1, 2, 3, 2, 3, 6 \rangle$, associated with their cost values when considering functions c_1, c_2 and c_3.

In order to simplify the algorithms' descriptions, in what follows will be assumed, with no loss of generality, that $\mathcal{P}_{is} = \emptyset$ and $\mathcal{P}_{ti} = \emptyset$, for any $i \in \mathcal{N}$.

p	$c_1(p)$	$c_2(p)$	$c_3(p)$
$\langle 1,2,3,6 \rangle$	40	2	20
$\langle 1,2,4,6 \rangle$	49	2	24.5
$\langle 1,2,5,6 \rangle$	53	2	26.5
$\langle 1,2,3,2,3,6 \rangle$	45	2	22.5

Table 1: Paths and loopless paths from 1 to 6 in $(\mathcal{N},\mathcal{A})$ and respective costs

3 The ranking optimal paths problem

In the general optimal path problem it is intended to determine a path p from s to t in $(\mathcal{N},\mathcal{A})$, such that $c(p)$ is optimal, that is, it is intended to $\text{opt}_{p\in\mathcal{P}}\{c(p)\}$, where "opt" stands for optimize which can be either minimize or maximize.

Given a positive integer $K>1$, the purpose of the enumeration of K optimal paths problem, or the K optimal paths problem, is to determine a set $\mathcal{P}_K=\{p_1,\ldots,p_K\}$ of paths, such that:

- $p_k \in \mathcal{P}$, for any $k\in\{1,\ldots,K\}$;
- $c(p_k)$ *better than* $c(p_{k+1})$, for any $k\in\{1,\ldots,K-1\}$;
- $c(p_K)$ *better than* $c(p)$, for any path $p\in\mathcal{P}-\mathcal{P}_K$;
- p_k is determined before p_{k+1}, for any $k\in\{1,\ldots,K-1\}$.

It should be noticed that when considering $K=1$, the optimal path problem is obtained. Thus, this is a generalization of the previous problem. Moreover, *better than* stands for "$\leq$" in minimization case and stands for "$\geq$" in maximization case.

According to the cost functions defined in section 2 for the network in Figure 1, the optimal path problems $\min\limits_{p\in\mathcal{P}}\{c_1(p)\}$, $\max\limits_{p\in\mathcal{P}}\{c_2(p)\}$ and $\min\limits_{p\in\mathcal{P}}\{c_3(p)\}$ could be formulated. These are usually known as shortest path problem, maximum capacity path problem and shortest path *per* capacity problem, [7], respectively, [9]. Obviously, other optimal path problems could be considered for different cost functions. For examples of different optimal path problems see [1, 4, 5, 10].

A problem similar to the ranking of K optimal paths is the enumeration of K optimal loopless paths. The goal of this problem is still the same; however the determined paths $p_1,\ldots,p_K$ are not allowed to have repeated nodes, that is, they have to be loopless.

Using $K=2$ and the network in Figure 1, the two optimal loopless paths for the three above-mentioned cost functions are $p_1=\langle 1,2,3,6\rangle$ and $p_2=\langle 1,2,4,6\rangle$. For the shortest loopless path problem, $c_1(p_1)=40$ and $c_1(p_2)=49$; for the maximum capacity loopless path problem, $c_2(p_1)=c_2(p_2)=2$; and finally, for the shortest *per* capacity unit loopless path problem, $c_3(p_1)=20$ and $c_3(p_2)=24.5$. (Notice that in the maximum capacity loopless problem all loopless paths from 1 to 6 have the same cost; therefore p_1 and p_2 can be either of them.)

4 Tree of K paths

The algorithms which will be presented for ranking the K optimal (loopless) paths between a pair of nodes in a given network are based on the construction of a "pseudo"-tree. This is not a tree in the usual sense since it can contain repeated nodes; however, if the same node is distinguished when belonging to different paths, it can be seen in such a way. Therefore, this "pseudo"-tree is often designated simply by tree of (loopless) paths. This "pseudo"-tree is defined by construction in the proof of Lemma 4.1, [8].

Lemma 4.1 *Given k, a positive integer, k paths between a pair of nodes in a network form a "pseudo"-tree.*
Proof: See [8].

To understand the paths tree notion it is important to notice that given k paths $q_1, \ldots, q_k$, defined between the same pair of nodes, each path q_i coincides with the previous ones, $q_1, \ldots, q_{i-1}$, in a subpath (otherwise q_i would coincide with one of the paths $q_1, \ldots, q_{i-1}$) from s to another node (which can be s itself and the subpath is then a null one). Among these nodes, the farthest from s (when concerning the number of intermediate nodes) is denoted by $d(q_i)$ and it is called the deviation node of q_i. By assumption $d(q_1) = s$.

In order to clarify these notions, let $q_1 = \langle 1, 2, 3, 6 \rangle$, $q_2 = \langle 1, 2, 5, 6 \rangle$ and $q_3 = \langle 1, 2, 4, 6 \rangle$ be three paths from $s = 1$ until $t = 6$ in the network represented in Figure 1. By assumption $d(q_1) = s = 1$. Moreover, path q_2 deviates from q_1 in node 2; therefore $d(q_2) = 2$. Similarly q_3 deviates from previous paths q_1 and q_2 in node 2; so $d(q_3) = 2$.

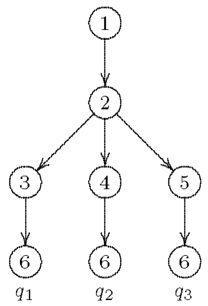

Figure 2: Tree of paths from 1 to 6 in $(\mathcal{N}, \mathcal{A})$

Finally it should be noticed that a loopless path is a special path (since it can not have repeated nodes); so, it is easy to see that K loopless paths also form an analogous tree of paths or, loopless paths tree as it is often called. The tree depicted in Figure 2 is the tree of loopless paths q_1, q_2 and q_3, once they do not contain cycles.

5 Deviation algorithm for ranking optimal loopless paths

It should be noticed that a tree of (loopless) paths is not directly related with any cost function. However, if such a function is considered, then an order in any set of (loopless) paths between two nodes of a network can be established, such as in the description of the K optimal (loopless) paths problem. Thus it is usual to refer to K optimal (loopless) paths tree whenever a cost function is considered.

In the K optimal (loopless) paths problem it is intended to compute the K optimal (loopless) paths tree and the general algorithm which will be proposed is based on this structure. However, most of the times a tree which "contains" this one has to be computed, that is, some non K optimal (loopless) paths are usually generated. In this section the k-th optimal loopless path from s to t will be denoted by p_k and it will be considered to be of the form $p_k = \langle s = v_1^k, v_2^k, \ldots, v_{\ell_k}^k = t \rangle$.

In what follows it is assumed the existence of some efficient algorithm for determining the optimal loopless path between a given pair of nodes, which will obviously depend on the considered cost function. If such an algorithm is known, loopless path p_1 can be computed and then some paths candidates for p_2, the second optimal loopless path, are generated. Like p_2, these candidate loopless paths separate from p_1 in one of its nodes. Thus, for each node v_i^1 a loopless path p is determined, such that it coincides with p_1 in $\mathrm{sub}_{p_1}(s, v_i^1)$ and only $\mathrm{sub}_p(v_i^1, t)$ has to be calculated. Since it is intended to determine loopless paths, no node can be repeated in p; therefore $\mathrm{sub}_p(v_i^1, t)$ can not contain any node in $\mathrm{sub}_{p_1}(s, v_i^1)$, except node v_i^1 itself. Moreover, once p separates from p_1 exactly in v_i^1, the first arc in $\mathrm{sub}_p(v_i^1, t)$ can not be (v_i^1, v_{i+1}^1). Finally, it should be reminded that it is intended to determine p_2; thus, the determined loopless "deviation" paths should be the *best*, under the above-mentioned constraints. For short, $p = \mathrm{sub}_{p_1}(s, v_i^1) \diamond q$ has to be the optimal loopless path such that $q \in \mathcal{P}_{v_i^1 t}$ does not contain any of the nodes in $\mathrm{sub}_{p_1}(s, v_{i-1}^1)$ and whose first arc is not (v_i^1, v_{i+1}^1). In order to calculate such a loopless path, those nodes and that arc are removed from the given network and only loopless path q has to be computed.

A possible approach to compute p could be to solve the optimal loopless path problem from v_i^1 to t in the modified network, concatenate it with $\mathrm{sub}_{p_1}(s, v_i^1)$ and, finally, restore the initial network. However, this approach is not valid for a general cost function, unless some adaptations are performed, as it will now be exemplified, with the network in Figure 1, where $s = 1$ and $t = 6$.

Let us consider the shortest path problem and let us assume that p_1 and p_2 have to be determined. Using some shortest path algorithm, for instance a labeling algorithm, [2], $p_1 = \langle 1, 2, 3, 6\rangle$ is calculated. After that, its nodes 1, 2 and 3 are analysed, in order to determine candidates for the second shortest path from 1 to 6 in $(\mathcal{N}, \mathcal{A})$. Let us consider the analysis of node 2, which means that the shortest loopless path with the form $p = \mathrm{sub}_{p_1}(1, 2) \diamond q = \langle 1, 2\rangle \diamond q$, where $q \in \mathcal{P}_{26}$ and its first arc is not $(2, 3)$, has to be computed. After removing node 1 and arc $(2, 3)$ from $(\mathcal{N}, \mathcal{A})$, the shortest loopless path from 2 to 6 is $q = \langle 2, 4, 6\rangle$; therefore $p = \langle 1, 2, 4, 6\rangle$, and $c_1(p) = c_1(\langle 1, 2\rangle) + c_1(q) = 10 + 39 = 49$. Notice that no paths are generated when analysing nodes 1 and 3, since the only arcs starting in these nodes are precisely the ones used in p_1, which can not be repeated. So, p is the loopless path p_2. Note that Yen's ranking loopless paths algorithm, [12, 13], has been briefly exemplified.

Let us now consider a different K optimal loopless paths problem, for instance, the shortest *per* capacity unit loopless path problem, also with $K = 2$. Still in this case $p_1 = \langle 1, 2, 3, 6\rangle$. After deleting node 1 there are two paths from node 2 to 6 whose first arc is not $(2, 3)$: $q_1 = \langle 2, 4, 6\rangle$ with cost $c_3(q_1) = 3.9$ and $q_2 = \langle 2, 5, 6\rangle$ with cost $c_3(q_2) = 2.15$. The best of these loopless paths is q_2; therefore, following the procedure used to exemplify the K shortest loopless paths problem, $p = \mathrm{sub}_{p_1}(1, 2) \diamond q_2 = \langle 1, 2, 5, 6\rangle$ with $c_3(p) = 26.5$, should be the best loopless path deviating from p_1 in 2. However, $p_2 = \langle 1, 2, 4, 6\rangle$, since $c_3(\langle 1, 2, 4, 6\rangle) = 24.5$; that is, the procedure above is not valid for this particular optimal path problem. Actually the described procedure is valid if the cost function satisfies (1), that is,

$$c(q) \geq c(q') \implies c(p \diamond q) \geq c(p \diamond q')\,, \tag{1}$$

for any paths p, q and q' such that $p \diamond q$ and $p \diamond q'$ can be defined. This happens for several cost functions, for instance, in the shortest path problem as well as in the maximum capacity path problem, as shown in Lemma 5.1 and Corollary 5.1.1.

Lemma 5.1

1. *Let us assume that $c(p \diamond q) = c(p) + c(q)$, for a given cost function c and for any paths p and q such that $p \diamond q$ is defined. Then the K optimal paths problem for $\mathrm{opt}_{p\in\mathcal{P}}\{c(p)\}$ satisfies (1).*
2. *Let us assume that $c(p \diamond q) = \min\{c(p), c(q)\}$, for a given cost function c and for any paths p and q such that $p \diamond q$ is defined. Then the K optimal paths problem for $\mathrm{opt}_{p\in\mathcal{P}}\{c(p)\}$ satisfies (1).*

Proof:

1. Let us assume that $c(p \diamond q) = c(p) + c(q)$, for any paths p and q such that $p \diamond q$ is defined, and let us suppose that (1) is not satisfied. Then, there should exist paths q and q' such that $c(q) \geq c(q')$ and $c(p \diamond q) < c(p \diamond q')$ for some path p.

   Since $c(p \diamond q) = c(p) + c(q)$ and $c(p \diamond q') = c(p) + c(q')$ it is easy to see that

   $$c(p \diamond q) < c(p \diamond q') \Leftrightarrow c(p) + c(q) < c(p) + c(q') \Leftrightarrow c(q) < c(q'),$$

   which contradicts $c(q) \geq c(q')$.

2. Let us now assume that $c(p \diamond q) = \min\{c(p), c(q)\}$, for every p and q such that $p \diamond q$ is defined, and that (1) is not verified. As in the last proof, there should exist q and q' such that $c(q) \geq c(q')$ and $c(p \diamond q) < c(p \diamond q')$ for some p.

   Three cases have to be considered: $c(p) \geq c(q) \geq c(q')$; $c(q) \geq c(p) \geq c(q')$ and $c(q) \geq c(q') \geq c(p)$.

   In the first one, $c(p \diamond q) < c(p \diamond q') \Leftrightarrow \min\{c(p), c(q)\} < \min\{c(p), c(q')\} \Leftrightarrow c(q) < c(q')$, which contradicts $c(q) \geq c(q')$.

   In the second case, $c(p \diamond q) < c(p \diamond q') \Leftrightarrow c(p) < c(q')$, which contradicts $c(p) \geq c(q')$.

   Finally, if $c(q) \geq c(q') \geq c(p)$ then $c(p \diamond q) < c(p \diamond q') \Leftrightarrow c(p) < c(p)$.

   It can then be concluded that (1) is satisfied. □

Corollary 5.1.1 *The K shortest paths problem and the K maximum capacity paths problem verify (1).*

Actually, simply determining q as the optimal loopless path in a given set of paths, the information about the cost of the initial subpath of p_1, that is $c(\mathrm{sub}_{p_1}(s, v_i^1))$, is not taken into account, although it may influence the total cost of loopless path p. Therefore, q has to be the optimal loopless path from v_i^1 to t, such that $c(\langle v_i^1 \rangle) = c(\mathrm{sub}_{p_1}(s, v_i^1))$ in the network modified as described above.

Let us now assume that loopless paths $p_1, \dots, p_{k-1}$ have been determined and the *best* candidates for p_j, with $j \geq k$, have been generated. Then p_k is the candidate path with an optimal cost. The new loopless paths to be generated from p_k, and candidates to p_j, with $j > k$, should deviate from p_k in one of its nodes; so, a similar procedure to the one used when considering p_1 will be outlined.

Let v_i^k be a p_k node to be analysed. The *best* loopless path deviating from p_k in v_i^k which has not been calculated yet should be generated, that is, a loopless path with the form $p = \mathrm{sub}_{p_k}(s, v_i^k) \diamond q$, where q does not contain any node in $\mathrm{sub}_{p_k}(s, v_{i-1}^k)$ neither the arc (v_i^k, v_{i-1}^k), not yet computed and such that $c(p)$ is the *best* under these conditions.

Two important issues regarding some differences between the analysis of p_1 and p_k must be specially studied. The first one concerns the nodes which should be analysed. Since p_k coincides with previous loopless paths $p_1, \dots, p_{k-1}$ from s until the deviation node $d(p_k)$, these nodes have already been analysed; therefore only nodes in $\mathrm{sub}_{p_k}(d(p_k), v_{\ell_k}^k)$ should be considered. The second issue also concerns the non recalculation of loopless paths, but when analysing the particular node $d(p_k)$. In fact, this node can have already been examined in order to generate loopless paths. Therefore, when examining $d(p_k)$, all arcs in $\mathcal{A}$ whose tail node is $d(p_k)$ and belonging to the loopless paths tree already constructed by the algorithm should be removed from $\mathcal{A}$. After these network changes q can be computed, being the optimal loopless path from v_i^k until t in the obtained modified network when considering $c(\langle v_i^k \rangle) = c(\mathrm{sub}_{p_k}(s, v_i^k))$. Finally the initial network should be restored.

As it was explained, several candidates for $p_1, \dots, p_K$ are computed when using this algorithm. To achieve this, a set X is used, where these candidate paths are stored until they are picked up in X, as being the loopless path in X with the *best* cost and then identified with some p_k.

From what has been described above, a general deviation algorithm for ranking K optimal loopless paths can be designed, which is outlined in Algorithm 1.

Algorithm 1 – *Ranking optimal loopless paths algorithm*

$p \longleftarrow$ optimal path from s to t in $(\mathcal{N}, \mathcal{A})$; $d(p) \longleftarrow s$; $X \longleftarrow \{p\}$; $k \longleftarrow 0$;
`While` $(X \neq \emptyset$ and $k < K)$ `Do`
  $k \longleftarrow k + 1$;
  $p_k \longleftarrow \langle v_1^k, \dots, v_{\ell_k}^k \rangle$; /* optimal loopless path in X */
  $X \longleftarrow X - \{p_k\}$;
  Remove nodes of loopless path $\mathrm{sub}_{p_k}(s, v_{\ell_k - 1}^k)$ from the network;
  Remove arcs $(d(p_k), i)$, $i \in \mathcal{N}$, of $p_1, \dots, p_{k-1}$ from the network;
  `For` $(v_i^k \in \{d(p_k), \dots, v_{\ell_k - 1}^k\})$ `Do`
    Remove arc (v_i^k, v_{i+1}^k), from the network;
    $q \longleftarrow$ optimal loopless path from v_i^k to t s.t. $c(\langle v_i^k \rangle) = c(\mathrm{sub}_{p_k}(s, v_i^k))$ in the modified network;
    $p \longleftarrow \mathrm{sub}_{p_k}(s, v_i^k) \diamond q$; $d(p) \longleftarrow v_i^k$; $X \longleftarrow X \cup \{p\}$;
    Remove node v_i^k from the network;
  `EndFor`
  Restore deleted nodes and arcs in the original network;
`EndWhile`

In order to clarify the algorithm's behaviour, the determination of the three shortest loopless paths *per* capacity will now be exemplified. It will still be considered the network $(\mathcal{N}, \mathcal{A})$ depicted in Figure 1 and $K = 3$. An algorithm for computing the shortest loopless path *per* capacity between two nodes of a given network will be assumed to be known (see, [7]).

The algorithm starts by computing the optimal loopless path from 1 to 6, which is $\langle 1,2,3,6\rangle$; thus $X = \{\langle 1,2,3,6\rangle\}$. In the first step of the algorithm $k = 1$ and $p_1 = \langle 1,2,3,6\rangle$, being this path picked up and deleted from X. Since $d(p_1) = s = 1$, nodes 1, 2 and 3 will be analysed; however, no new loopless path is generated from node 1, since the only arc of $(\mathcal{N},\mathcal{A})$ starting in 1 is the one used in p_1. Yet, when analysing 2, node 1 and arc $(2,3)$ are deleted from $(\mathcal{N},\mathcal{A})$ and the shortest loopless path *per* capacity from 2 to 6 such that $c(\langle 2\rangle) = 5$ is determined; that is, $q = \langle 2,4,6\rangle$. Thus $p = \langle 1,2\rangle \diamond \langle 2,4,6\rangle = \langle 1,2,4,6\rangle$ is stored in X. Analysing 3, node 2 and arc $(3,6)$ are also deleted from the network and no loopless paths from 3 to 6 exist in the remaining network; so $X = \{\langle 1,2,4,6\rangle\}$. Once again there is only one loopless path in X; so, in the following step of the algorithm, $k = 2$ and $p_2 = \langle 1,2,4,6\rangle$, being p_2 removed from X. Since $d(p_2) = 2$, nodes 2 and 4 have to be analysed. When analysing 2, node 1 and arcs $(2,3)$ and $(2,4)$ are deleted and the shortest loopless path *per* capacity unit from 2 to 6 will be $\langle 2,5,6\rangle$. After this, $X = \{\langle 1,2,5,6\rangle\}$ and then 4 is analysed but no loopless path is generated. Finally in the third step, $k = 3$, path $p_3 = \langle 1,2,5,6\rangle$ and once there are no more loopless paths from 1 to 6 in the network the algorithm stops.

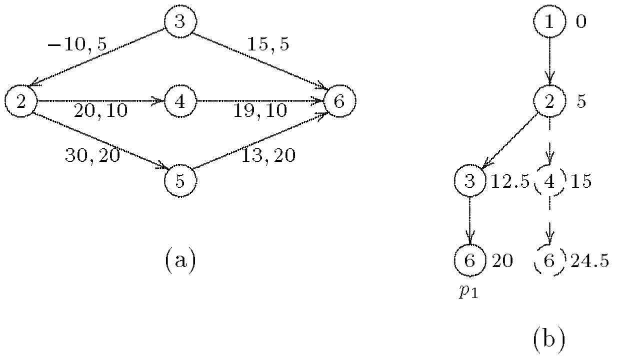

Figure 3: Ranking loopless paths algorithm, $k = 1$: (a) modified network; (b) determined loopless paths tree

The loopless paths tree which is being built and the modified networks where new loopless paths are computed in steps $k = 1$ and $k = 2$, are depicted in Figures 3 and 4, respectively. The dashed branches in the loopless paths trees denote loopless paths belonging to X but not yet chosen as some p_k. Moreover, the real value close to each node i in the loopless paths tree denotes the cost of the loopless path in this tree, from the initial node until node i.

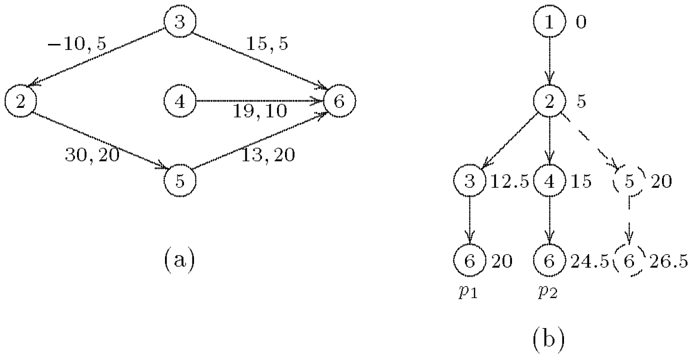

Figure 4: Ranking loopless paths algorithm, $k = 2$: (a) modified network; (b) determined loopless paths tree

6 Deviation algorithm for ranking optimal paths

In the general K optimal paths problem, some unconstrained paths can be determined; so, nodes can be repeated in the determined paths. As it was seen in section 4, K paths, as well as K loopless paths, form a tree structure, known as tree of K paths and tree of K loopless paths. Therefore, the algorithm designed in the previous section can be adapted for ranking unconstrained paths, as will now be described. In this section $p_k = \langle v_1^k, \dots, v_{\ell_k}^k \rangle$ will denote the k optimal path from s to t for some $k \in \{1, \dots, K\}$.

Let us consider the analysis of a node v_i^k belonging to some path p_k, intending now to compute the *best* path deviating from p_k exactly in v_i^k and which has not been determined yet. Such a path has the form $p = \text{sub}_{p_k}(s, v_i^k) \diamond q$, where q is the path with *best* cost in $\mathcal{P}_{v_i^k t}$ assuming that $c(\langle v_i^k \rangle) = c(\text{sub}_{p_k}(s, v_i^k))$ and whose first arc is not of the form (v_i^k, x), belonging to the tree of paths already built. It is important to notice that since no constraints are imposed on the path definition, nodes in $\text{sub}_{p_k}(s, v_{i-1}^k)$ are allowed to appear also in q; thus they will not be removed from the network. Furthermore, since v_i^k can be repeated in q, those arcs (v_i^k, x) can be used again and therefore they can not be deleted from the $(\mathcal{N}, \mathcal{A})$. Path q will be of the form $q = (v_i^k, y) \diamond r$, $r \in \mathcal{P}_{yt}$, where (v_i^k, y) is its first arc, therefore subjected to some constraints. In order to compute q, a new node, denoted by $v_i^{k\prime}$, will be added to the network. Node $v_i^{k\prime}$ will only be used as the first node of q, therefore arc $(v_i^{k\prime}, j)$ is also added to $\mathcal{A}$, for every $(v_i^k, j) \in \mathcal{A}$ allowed to be the first arc of q. Since $v_i^{k\prime}$ corresponds to node v_i^k in the original network, the generated path will be $\text{sub}_{p_k}(s, v_i^k) \diamond (v_i^k, y) \diamond r$.

This adaptation is described in Algorithm 2.

Algorithm 2 – *Ranking optimal paths algorithm*

```
p ⟵ optimal path from s to t in (N, A);  d(p) ⟵ s;  X ⟵ {p};  k ⟵ 0;
While (X ≠ ∅ and k < K) Do
    k ⟵ k + 1;
    p_k ⟵ ⟨v_1^k, ..., v_{ℓ_k}^k⟩;    /*  optimal path in X   */
    X ⟵ X − {p_k};
    Add node d(p_k)' to the network;
    For ((d(p_k), i) ∈ A not in p_1, ..., p_k) Do Add arc (d(p_k)', i) to the network;
    q ⟵ optimal path from d(p_k)' to t s.t. c(⟨d(p_k)'⟩) = c(sub_{p_k}(s, d(p_k)));
    q' ⟵ path equivalent to q in the original network;
    p ⟵ sub_{p_k}(s, d(p_k)) ◇ q';  d(p) ⟵ d(p_k);  X ⟵ X ∪ {p};
    Restore the original network;
    For (v_i^k node of p_k following d(p_k)) Do
        Add node v_i^k' to the network;
        For ((v_i^k, j) ∈ A − {v_i^k, v_{i+1}^k}) Do Add (v_i^k', j) to the network;
        q ⟵ optimal path from v_i^k' to t s.t. c(⟨v_i^k'⟩) = c(sub_{p_k}(s, v_i^k));
        q' ⟵ path equivalent to q in the original network;
        p ⟵ sub_{p_k}(s, v_i^k) ◇ q';  d(p) ⟵ v_i^k;  X ⟵ X ∪ {p};
        Restore the original network;
    EndFor
EndWhile
```

In what follows this algorithm will be exemplified when determining the two shortest paths *per* capacity unit from 1 to 6 in the network depicted in Figure 1.

The algorithm starts by calculating the optimal path from 1 to 6, which is $\langle 1, 2, 3, 6 \rangle$; thus $X = \{\langle 1, 2, 3, 6 \rangle\}$. In the first step of the algorithm, $k = 1$ and $p_1 = \langle 1, 2, 3, 6 \rangle$ and this path is picked up and removed from X. Since $d(p_1) = s = 1$, nodes 1, 2 and 3 will be analysed. When analysing 2, node $2'$ and arcs $(2', 4)$ and $(2', 5)$ are added to $(\mathcal{N}, \mathcal{A})$ and the shortest path *per* capacity unit from $2'$ to 6 such that $c(\langle 2' \rangle) = 5$ is determined. Then $q = \langle 2', 4, 6 \rangle$, $X = \{\langle 1, 2, 4, 6 \rangle\}$ and the initial network is restored. Now considering node 3, node $3'$ and arc $(3', 2)$ are created and the shortest path *per* capacity unit from $3'$ to 6 such that $c(\langle 3' \rangle) = 12.5$ is $q = \langle 3', 2, 3, 6 \rangle$. Thus $p = \langle 1, 2, 3, 2, 3, 6 \rangle$ is stored in X. At this moment $X = \{\langle 1,2,4,6\rangle, \langle 1,2,3,2,3,6\rangle\}$ and in the second step of this algorithm $p_2 = \langle 1,2,3,2,3,6\rangle$ is chosen in X. In this step nodes 2 and 3 will be examined since $d(p_2) = 2$; after that $X = \{\langle 1,2,4,6\rangle, \langle 1,2,3,2,4,6\rangle, \langle 1,2,3,2,3,2,3,6\rangle\}$ and the algorithm ends.

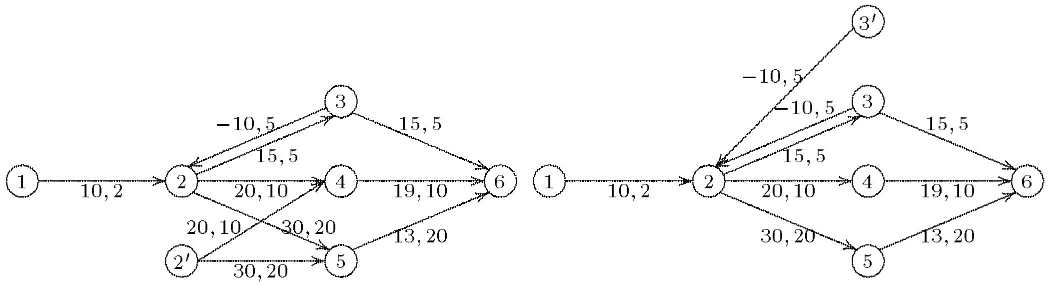

Figure 5: Modified networks used by ranking paths algorithm when $k = 1$ and $k = 2$

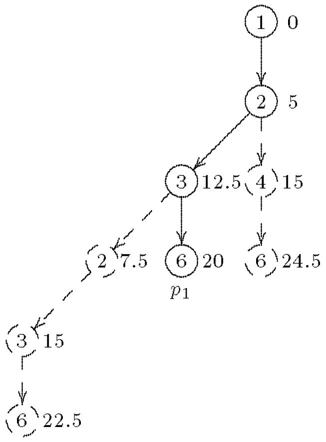

Figure 6: Paths tree determined by ranking paths algorithm when $k = 1$

Analogously to the loopless case, Figure 5 represents the modified networks used by the algorithm in steps $k = 1$ and $k = 2$, while Figures 6 and 7 represent the paths tree built by the algorithm when $k = 1$ and $k = 2$, respectively. Dashed branches correspond to determined paths not yet picked up in X and the real value close to each node in the paths tree denotes the cost of the path in this tree, from s until that node.

7 Computational complexity

In this section the theoretical computational complexity for the presented algorithms, in a worst-case anaylis, will be briefly studied.

Let us first consider the K optimal loopless paths algorithm. In this algorithm, K loopless paths are picked up in the candidates set, X. For each one of those chosen loopless paths, its nodes from the deviation until the terminal node have to be analysed. Thus, when considering the worst-case for this algorithm, all the n nodes in the network have to be analysed, and for each one of them an optimal loopless path problem has to be solved. Therefore, assuming that solving such a problem demands $c(n)$ operations, the computational complexity of the proposed ranking algorithm, for the loopless case, is $\mathcal{O}(Kn\, c(n))$.

When the unconstrained case is concerned, if there is always a loopless path which is an optimal path, that is, if any optimal path problem has a loopless path as optimal solution, then at the most n nodes are analysed for every path p_k and again the complexity order for this problem is $\mathcal{O}(Kn\, c(n))$. Otherwise, the

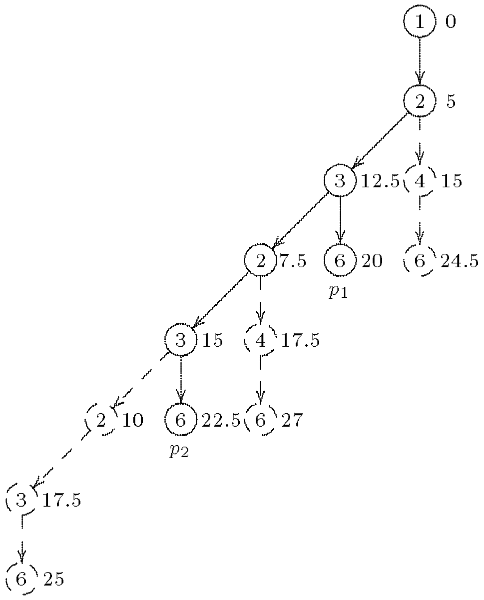

Figure 7: Paths tree determined by ranking paths algorithm when $k = 2$

number of nodes to analyse when considering some path p_k appears not to be known in advance and the computational order for this algorithm can not be estimated.

References

[1] Ravindra K. Ahuja. Minimum cost-reliability ratio path problem. *Computers and Operations Research*, 15(1):83–89, 1988.

[2] Ravindra K. Ahuja, Thomas L. Magnanti and James B. Orlin. *Network Flows : Theory, Algorithms and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[3] Y. L. Chen. Finding the k quickest simple paths in a network. *Information Processing Letters*, 50:89–92, 1994.

[4] 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.

[5] P. Hansen, G. Shtorchi and T. Vovor. Paths with minimum range and ratio of arc lengths. *Discrete Applied Mathematics*, 78:89–102, 1997.

[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*, 18:401–405, 1972.

[7] E. Q. V. Martins. An algorithm to determine a path with minimal cost/capacity ratio. *Discrete Applied Mathematics*, 8:189–194, 1984.

[8] E. Q. V. Martins and M. M. B. Pascoal. A new implementation of Yen's ranking loopless paths algorithm. Submitted, 2000.
(`http://www.mat.uc.pt/~eqvm/cientificos/investigacao/Artigos/new_yen.ps.gz`).

[9] E. Q. V. Martins, M. M. B. Pascoal, D. M. L. D. Rasteiro and J.L.E. Santos. The optimal path problem. *Investigação Operacional*, 19:43–60, 1999.
(`http://www.mat.uc.pt/~eqvm/cientificos/investigacao/Artigos/opath.ps.gz`).

[10] E. Q. V. Martins and J. L. E. Santos. An algorithm for the quickest path problem. *Operations Research Letters*, 20:195–198, 1997.

[11] R. G. Murty. An algorithm for ranking all the assignments in increasing order of cost. *Operations Research*, 16:682–687, 1968.

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

[13] J. Y. Yen. Shortest path network problems, 1975. (Mathematical Systems in Economics, Heft 18), Hain, (1975), Meisennheim am Glan.