LABELING ALGORITHMS FOR RANKING SHORTEST PATHS

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

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

March 1999
Revised July and October 1999

Abstract: In this paper the ranking of shortest paths problem is viewed as a generalization of the shortest path problem, since the K shortest paths have to be computed. Finitness and boundness are studied as well as conditions are established in order that an Optimality Principle generalization is satisfied. Under these conditions the K shortest paths can be determined by a natural generalization of forms of the labeling algorithm for the shortest path problem. This generalization is the main subject of the paper.

Keywords: network, path, path cost, paths ranking, optimality principle, labeling.

1 Introduction

In this paper the ranking of shortest paths is considered as a generalization of the well known shortest path problem, [5, 6, 7, 8, 9, 10, 12], since several paths must be determined. In fact, in the ranking of shortest paths problem, for a given integer K ($K \geq 1$), the K shortest paths between a given pair of nodes have to be listed by nondecreasing order of their costs. Sometimes K depends on some constraints that must be satisfied by the paths; for example, in the constrained shortest path problem it is intended to compute all the paths between a given pair of nodes whose cost is not greater than a given value, [1].

Two classes of ranking shortest paths problems are usually considered depending on the existence or non existence of some constraints in the path definition. For example, in the ranking of shortest loopless paths, only paths without repeated vertices (and arcs) are allowed in the final solution, [13, 15, 20, 24].

In this paper we are concerned with the unconstrained problem. Finitness and boundness are defined and studied as well as conditions are established in order that the problem satisfies an Optimality Principle generalization. Under these conditions, the problem can be solved by a natural generalization of forms of the labeling algorithm for the classical shortest path problem. This generalization is the main subject of this paper.

2 Definitions and Notation

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 or vertices and $\mathcal{A}=\{a_1,\ldots,a_m\}$ is a finite set whose elements are called arcs. Each arc a_k is a pair (v_i,v_j) of nodes — so, an arc a_k may also be represented by (v_i,v_j). $(\mathcal{N},\mathcal{A})$ is a directed (undirected) network if (v_i,v_j) are ordered (unordered) pairs, for any $(v_i,v_j)\in\mathcal{A}$. Throughout and with no loss of generality $(\mathcal{N},\mathcal{A})$ is a directed network. To simplify the notation, sometimes it will be used $\{1,\ldots,n\}$ to mean $\{v_1,\ldots,v_n\}$ and $\{1,\ldots,m\}$ to mean $\{a_1,\ldots,a_m\}$; so, i will be used to denote the node v_i and (i,j) will be used to denote the arc $a_k=(v_i,v_j)$.

Let i and j be two nodes of $(\mathcal{N},\mathcal{A})$. A path from i to j in $(\mathcal{N},\mathcal{A})$ is an alternating sequence of nodes and arcs, of the form $\langle i=v'_1,a'_1,v'_2,\ldots,a'_{r-1},v'_r=j\rangle$, where:

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

Let $\mathcal{P}_{ij}$ denote the set of paths that can be defined in $(\mathcal{N},\mathcal{A})$ from i to j, for any two nodes $i,j\in\mathcal{N}$ and let $\bigcup_{i,j\in\mathcal{N}}\mathcal{P}_{ij}$ denote the set of paths that can be defined in $(\mathcal{N},\mathcal{A})$. For any arc $(i,j)\in\mathcal{A}$, let c_{ij} be a real number associated with arc (i,j); c_{ij} is said to be the (i,j) arc value or the (i,j) arc cost. Let $c:\bigcup_{i,j\in\mathcal{N}}\mathcal{P}_{ij}\longrightarrow\mathbb{R}:p\longrightarrow c(p)=\sum_{(i,j)\in p}c_{ij}$ be a given function that assigns a real value $c(p)$ to each path p that is defined in $(\mathcal{N},\mathcal{A})$ — $c(p)$ is said to be the p path value or the p path cost.

A path p from i to j in $(\mathcal{N},\mathcal{A})$ is said to be loopless if no node (and no arc) is repeated in p. A cycle is a path from some node to itself, where all the nodes are different except the first which also is the last one. A cycle C in $(\mathcal{N},\mathcal{A})$ is said to be a negative cycle *if and only if* $\sum_{(i,j)\in C}c_{ij}<0$, i.e., the sum of its arcs cost is negative; C is said to be an absorbent cycle *if and only if* $\sum_{(i,j)\in C}c_{ij}\leq 0$, i.e., the sum of its arcs cost is not positive.

In what follows, let s and t denote two given nodes of $(\mathcal{N},\mathcal{A})$. With no loss of generality it will be assumed that $\mathcal{P}_{si}\neq\emptyset$ and $\mathcal{P}_{it}\neq\emptyset$ hold for any node $i\in\mathcal{N}$. Moreover $\mathcal{P}$ will be used to denote $\mathcal{P}_{st}$.

In the ranking of shortest paths problem, for a given integer K ($K\geq 1$), a given network $(\mathcal{N},\mathcal{A})$ and two 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}.

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.

Definition 1 – *A path is said to be finite if its number of nodes (and arcs) is finite.*

Theorem 1 – *Let us assume that $\mathcal{P}_{si}\neq\emptyset$ and $\mathcal{P}_{it}\neq\emptyset$, for any $i\in\mathcal{N}$. 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 – See [16]. ∎

Definition 2 – *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. ∎

In order to assure the finiteness of the K shortest paths problem, it will be assumed the non existence of negative cycles in $(\mathcal{N}, \mathcal{A})$. Moreover, to avoid that the last solutions are formed by the same path and one more turn around the same cycle it will be assumed that $c(C) > 0$ holds for any cycle C in $(\mathcal{N}, \mathcal{A})$; that is, it is assumed the non existence of absorbent cycles in $(\mathcal{N}, \mathcal{A})$.

3 Algorithms based on the Optimality Principle

A general optimal path problem can be solved by the labeling algorithm as long as the Optimality Principle is satisfied, [17]. The Optimality Principle asserts that *there is an optimal path $p^\star$ from s to t that is formed by optimal sub–paths from s to every intermediate node of $p^\star$.*

In the case of the classical shortest path problem it could be proved that it satisfies the Optimality Principle if and only if *there are no negative cycles in the network*. For $K > 1$, the Optimality Principle asserts that *for any $k \geq 1$, the $k^{\underline{\text{th}}}$ shortest path p_k, from s to t, is formed by $\xi_i{}^{\underline{\text{th}}}$ shortest paths, from s to every intermediate node i of p_k, and $\xi_i \leq k$.* Notice that the Optimality Principle is still valid for paths from s to any intermediate node of p_k.

In Theorem 3 conditions are established in order that the Optimality Principle is verified for the ranking shortest paths problem.

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 $i \in \mathcal{N}$. Then the $k^{\underline{\text{th}}}$ shortest path p_k, from s to t, is formed by $\xi_j{}^{\underline{\text{th}}}$ shortest paths, from s to every intermediate node j of p_k, and $\xi_j \leq k$.*

Proof – See [16]. ∎

A class of ranking shortest paths algorithms is supported by theorem 3. Dreyfus, [11], was the first one to propose an algorithm in this class. His 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. Also in this class of algorithms we can consider the natural labeling algorithm generalization for the shortest path problem. Shier was the first one to propose forms of this algorithm in this subclass [21, 22, 23]. Yet in the class of algorithms supported by the Optimality Principle, are included Martins' algorithm, [14], and all its improvements, [2, 3, 4, 18].

Labeling algorithm will be presented in the next subsection.

3.1 Labeling Algorithm

As all the forms of the shortest path labeling algorithm, their generalizations for the K shortest paths problem can be divided into two sets: the label correcting algorithms and the label setting algorithms, [6]. While label correcting algorithms determine the set $\mathcal{P}_K = \{p_1, \ldots, p_K\}$ only in the last step of the algorithm, in the label setting algorithms each path p_k is being determined throughout the computations. That is, while in the label setting algorithms it is well known that a path is the following one in $\mathcal{P}_K$ as soon as a new path from s to t is determined, in label correcting algorithms this is known only at the end of the algorithm.

Algorithm 1 – **General form of the labeling K shortest paths algorithm.**

/* X — subset of the set of nodes. */

$\{\pi_i^1, \ldots, \pi_i^K\} \longleftarrow \{+\infty, \ldots, +\infty\}$, **for all** node $i \in \mathcal{N}$
$\pi_s^1 \longleftarrow 0$; π_s^1 is *unused*
$X \longleftarrow \{s\}$
while $X \neq \emptyset$
do begin
  $i \longleftarrow$ node of X
  $X \longleftarrow X - \{i\}$
  for each arc $(i, j) \in \mathcal{A}$ **do**:
    for each $k \in \{1, \ldots, K\}$ such that π_i^k is *unused* and finite
    do begin
      π_i^k is *used*
      if $(\pi_i^k + c_{ij} < \max\{\pi_j^1, \ldots, \pi_j^K\})$
      then begin
        $\ell \longleftarrow$ order of $\max\{\pi_j^1, \ldots, \pi_j^K\}$ in $\{\pi_j^1, \ldots, \pi_j^K\}$
        $\pi_j^\ell \longleftarrow \pi_i^k + c_{ij}$; $\xi_j^\ell \longleftarrow i$; $\theta_j^\ell \longleftarrow k$
        π_j^ℓ is *unused*
        if $j \notin X$ **then** $X \longleftarrow X \cup \{j\}$
      end
    end
  end
Find the K shortest paths /* By nondecreasing order */

The forms of the labeling algorithm have in common the use of one or more labels assigned to each node. While in the shortest path problem there is a single label assigned to each node i, in the K shortest paths problem we may have a single label or several labels assigned to each node. In the former case the label of node i is formed by three K–tuple: π_i, ξ_i and θ_i. While π_i^k, the $k^{\underline{\text{th}}}$ component of π_i, is the cost of a path from s to node i, the $k^{\underline{\text{th}}}$ component of ξ_i and θ_i, respectively ξ_i^k and θ_i^k, is the node before i in that path and its position in ξ_i, respectively. Except π_s^1 that is initialized as zero, π_i^k is initialized as $+\infty$ for any $k \in \{1, \ldots, K\}$ and for any $i \in \mathcal{N}$. When it was tried to improve components of π_j from π_i^k, $i \neq j$, this component of π_i is said to be *used*; otherwise it is said to be *unused*.

Algorithm 1 is the general form of the K shortest paths labeling algorithm. It may suggest that the three K–tuple labeling process is not efficient. Among other reasons will be pointed out the following:

1. It may happen that some components of some labels remain $+\infty$ throughout the algorithm. This will happen if and only if the number of paths from s to some node i is less than K. In this case, it is obvious that the memory space is not being efficiently used, since the n K-tuples form a sparse $(n \times K)$ matrix.

2. Let i be some node picked out on X in the **while** loop of the algorithm and let us assume that the label of i is updated later on. So, i is included again in X and when picked out on X some calculations can not be executed again for some finite values of π_i^k – those which were *used* before the updating of π_i. So, a *flag* associated with each node i can be used; the flag of i would be a K–tuple whose $k^{\underline{\text{th}}}$ component is assigned with true if and only if π_i^k has already been previously used to attempt to label other nodes.

Algorithm 2 – **General form of the single label K shortest paths algorithm.**

```
/*
      X        —   subset of N*, the enlarged set of nodes.
      count_i  —   number of paths that were determined from s to i.
      elm      —   number of elements of N*.
                                                                   */
count_i ⟵ 0, for all i ∈ N
elm ⟵ 1
h(elm) ⟵ s; h^{-1}(s) ⟵ {elm}; π_elm ⟵ 0
X ⟵ {elm}
while (X ≠ ∅)
do begin
      k ⟵ some element of X
      X ⟵ X − {k}
      i ⟵ h(k)
      for each arc (i,j) ∈ A
      do begin
            if (count_j = K)      /* By the Optimality Principle no more than K */
            then begin            /* paths have to be determined from s to j */
                  ℓ ⟵ element of h^{-1}(j) for which π_ℓ = max{π_x | x ∈ h^{-1}(j)}
                  if (π_k + c_ij < π_ℓ)
                  then begin
                        π_ℓ ⟵ π_k + c_ij; ξ_ℓ ⟵ k
                        if ℓ ∉ X then X ⟵ X ∪ {ℓ}
                  end
            end
            else begin            /* (count_j < K) */
                  elm ⟵ elm + 1
                  π_elm ⟵ π_k + c_ij; ξ_elm ⟵ k
                  count_j ⟵ count_j + 1
                  h(elm) ⟵ j; h^{-1}(j) ⟵ h^{-1}(j) ∪ {elm}
                  X ⟵ X ∪ {elm}
            end
      end
end
Find the K shortest paths        /* By nondecreasing order */
```

3. The maximum component of the cost label of a node j has to be computed each time a node i is picked out on X and for each finite and *unused* π_i^k; this operation implies a really large amount of computational effort. Keeping sorted each π_i is also very hard since each change in π_i will imply rearranging backwards ξ_j and θ_j until π_s.

Computationally it seems to be really difficult to deal with the three K–tuple labels used in the general form of the labeling K shortest paths algorithm. However, a different version of this algorithm can be designed where each node is assigned with several labels. Basically, each finite component of the cost label of node i is considered as a label of node i itself.

From now on, this form will be used in the description of labeling K shortest paths algorithms. To simplify the notation (and because it results computationally more efficient), the set of nodes will be enlarged by the adding of new nodes and in such a way that a correspondence is established between the set of the labels and the enlarged set of nodes. That is, each node is assigned with a single label. A function $h : \mathbb{N} \longrightarrow \mathcal{N}$ has to be defined in order to make possible to assign a single label to some natural number, i.e., to some well determined node. Since some nodes in the original set may be assigned with more than a single label, the h function has no inverse; however and to make possible the determination of all the natural numbers corresponding to the same node we will define $h^{-1}(i) = \{k \in \mathbb{N} \mid h(k) = i\}$. Note that the set $h^{-1}(i)$ can be easily computed for any $i \in \mathcal{N}$ if it is viewed as a linked list.

Algorithm 2 is the general form of the K shortest paths labeling algorithm, where the set of nodes is enlarged in order that each node is assigned with a single label. Note that the enlarged set of nodes has at most nK elements.

Since $\mathcal{P}_K$ can only be determined in the last step of the algorithm, it also does not work when K is not known in advance. By the same reason and also by the Optimality Principle, the determination of $\max\{\pi_x \mid x \in h^{-1}(j)\}$ for each arc (i, j), each time that some k is picked out on X, allows that the number of labels assigned to each node needs not to be greater than K.

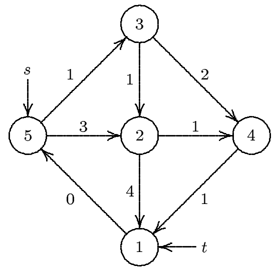


Figure 1: **Network to exemplify algorithms**

3.1.1 Label Correcting Algorithm

Since the general labeling algorithm has temporary labels, the general form for the label correcting algorithm is the one stated for the general case, that is, the form described in Algorithm 1 (or the one described in Algorithm 2). When there is no freedom to choose the node i in X (or the element k in X) and no freedom in adding a node j to X (or an element elm to X) a specific form is designed. For example, X may be implemented as a **FIFO** linked list what results in the **BFM** form — the generalization of the Bellman–Ford–Moore form, [5, 6, 19], for the shortest path problem.

Notice that the **BFM** form is similar to Algorithm 1 or Algorithm 2 (which depends on the use of a K-tuple or K single labels). For instance, relating to Algorithm 2, the one which uses K single labels, the only difference is that k is allways the first element of X and ℓ is allways placed in the last position of X.

The **BFM** form of the label correcting algorithm will be exemplified with the network represented in figure 1. For this purpose, the set of arcs will be considered arranged in such a way that $\mathcal{A} = \{a_1, \ldots, a_m\}$ and $a_k = (i, j) < a_\ell = (i', j')$ if and only if $i < i'$ or $i = i'$ and $j < j'$. Furthermore, in order to increase the readability of the example, the function h as well as h^{-1} and elm will not be used. As a consequence a node i will be primed with so many primes as the number of paths that were determined from s to i.

The ranking will be executed for $K = 2$ paths from $s = 5$ to $t = 1$. The obtained results are depicted in figure 2 (the first seven steps) and in figure 3 (the optimal solution with all the determined labels). Each circle is associated with a well determined path which is defined from $s = 5$ until the node with primes within the circle. Close to each circle we find a number which represents the cost of this path; the ξ and θ are graphically represented by the arrows. The variable $count$ is represented by the array in the bottom of each tree.

Notice that in step 7, we try to improve labels from node $2''$, that is, nodes 1 and 4. For node 1, since $\pi_{1'} = 7$, $\pi_{1''} = 5$ and $\pi_{2''} + c_{21} = 6$, node $1'$ will be removed and replaced by $1'''$ and $\pi_{1'''} = 6$. Similarly, node $4'$ is replaced by node $4'''$ and $\pi_{4'''} = 3$.

Dashed circles represent labels that had been computed and removed later on. We must notice the dashed subtree rooted at label $2'$; all the labels in this subtree were removed only after being picked out on X and being used to determine labels which will be improved later on.

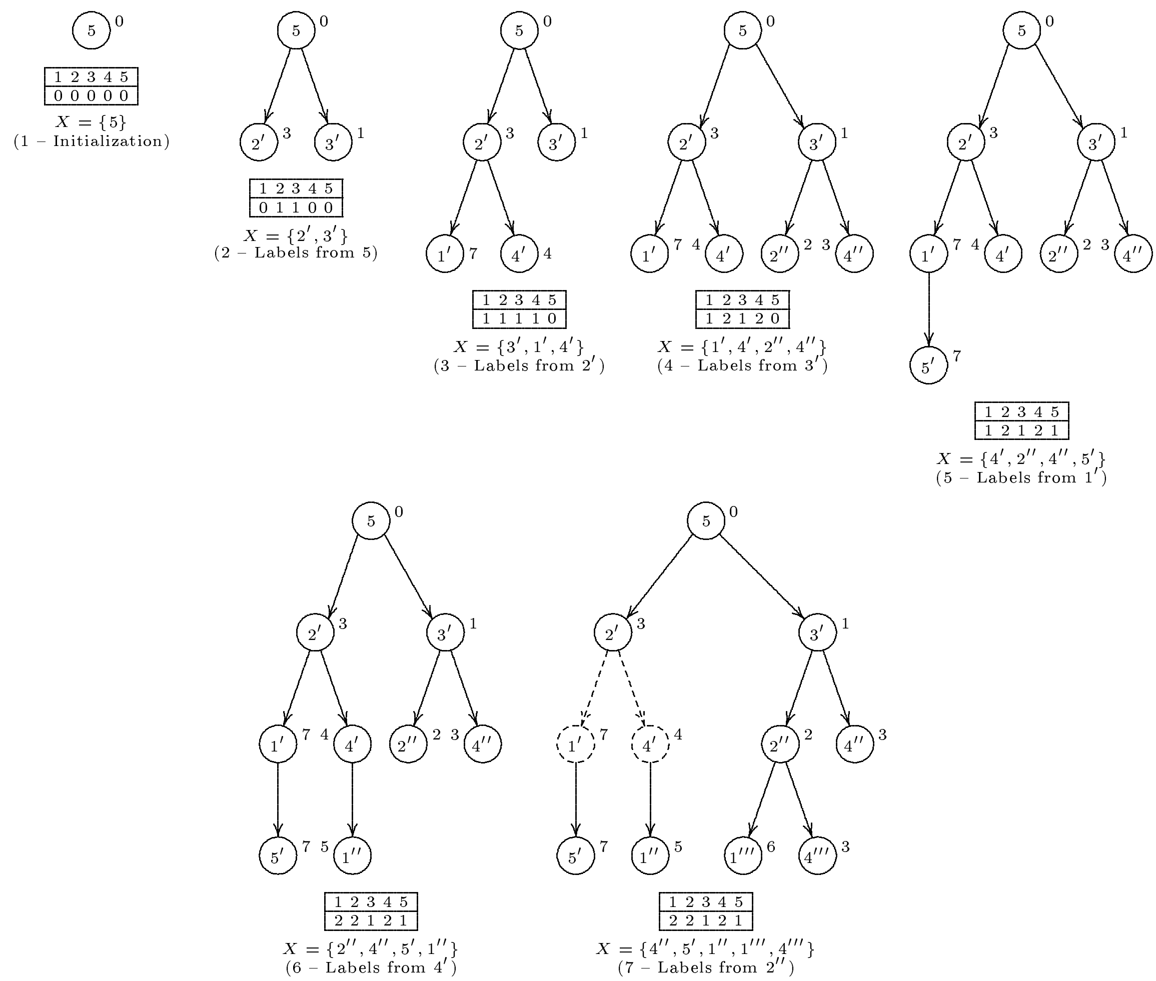


Figure 2: **The first seven steps of the BFM generalization form**

From the example it is immediate to conclude that the **BFM** form of the label correcting algorithm is not really efficient; we can even say that it works like a *brute force* method. Notice that in the brute force method all the possibilities are tested what implies the nonfiniteness of the algorithm unless the number of paths is finite (that is in acyclic networks). So, to assure finiteness of the brute force method for general networks, we use the Optimality Principle considering the K shortest paths from s to any node i, such as in label correcting algorithm.

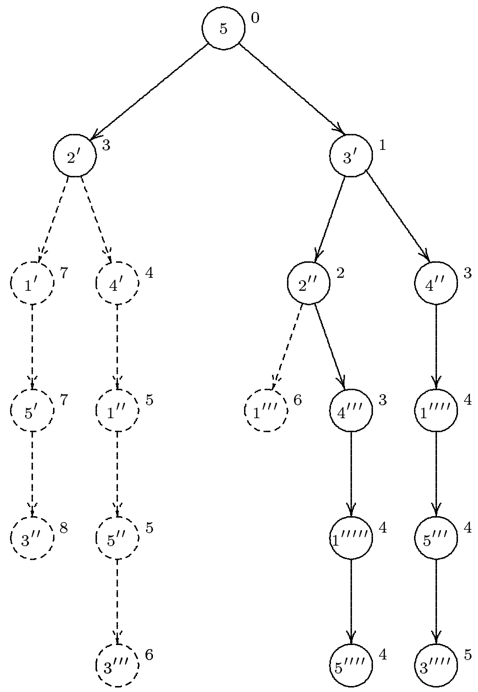
We must also notice that the theoretical complexity of this algorithm is not polynomial for a worst case analysis since it is not possible to establish a polynomial upper bound for the number of labels that are computed for each node.

We must notice that both forms of the algorithm, i.e., the form where each node is assigned with a single label and the form where each node is assigned with K labels, are coincident when $K = 1$.

Other forms of the label correcting algorithm could be designed, either with a K–tuple label assigned to each node or manipulating X in different ways. However, computational experiments have to be done in order to conclude about the efficiency of each one of the possible forms.

3.1.2 Label Setting Algorithm

The efficiency of the forms of the label correcting algorithm can be easily improved. In fact we may think to manipulate X in such a way that the label of the element picked out on X is permanent, such as in Dijkstra's algorithm for the shortest path problem, [10]. So, let us assume that $c_{ij} \geq 0$, for all $(i, j) \in \mathcal{A}$. Since for $K = 1$ the algorithm 3 is the classical Dijkstra's algorithm for the single shortest path problem, it is immediate that algorithm 3 is its generalization for the K shortest paths problem; so, it is a label setting algorithm.



$X = \emptyset$

Figure 3: **Labels obtained with the BFM form to determine the $K = 2$ shortest paths**

Notice that $\mathcal{P}_K$ is being computed with the execution of the algorithm and not as its last step. Moreover, whenever an element k is picked out on X for which $h(k) = t$ a new path is added to $\mathcal{P}_K$.

A different form of the algorithm could be designed. Namely, keeping in X only the K shortest paths from s to i, for any $i \in \mathcal{N}$. However, this form would be less efficient in terms of the execution time since keeping the K shortest paths from s to all $i \in \mathcal{N}$ is a very hard task (at least for significant values of K); the described form is more efficient with respect to the execution time but more labels are assigned what implies less efficiency in terms of the necessary memory space.

The theoretical computational complexity of this form can be established, considering a worst case

Algorithm 3 – **Generalization of Dijkstra's algorithm, with single labels.**

```
/*
      X      —  subset of N*, the enlarged set of nodes.
    count_i  —  number of paths that were determined from s to i.
     elm     —  number of elements of N*.
                                                              */
count_i ⟵ 0, for all i ∈ N − {s}; count_s ⟵ −1
elm ⟵ 1
h(elm) ⟵ s; h^{-1}(s) ⟵ {elm}; π_elm ⟵ 0
X ⟵ {elm}
P_K ⟵ ∅
While (count_t < K) and (X ≠ ∅)
do begin
     k ⟵ element of X such that π_k ≤ π_x, for any x ∈ X
     X ⟵ X − {k}
     i ⟵ h(k)
     count_i ⟵ count_i + 1
     if (i = t)
     then begin
             p ⟵ path from 1 to k
             P_K ⟵ P_K ∪ {h(p)}        /*   p_{count_t} ⟵ h(p)    */
          end
     if (count_i ≤ K)        /*    By the O. P. K paths have to be computed at most.    */
     then begin
             for each arc (i, j) ∈ A
             do begin
                   elm ⟵ elm + 1
                   π_elm ⟵ π_k + c_ij; ξ_elm ⟵ k
                   h(elm) ⟵ j; h^{-1}(j) ⟵ h^{-1}(j) ∪ {elm}
                   X ⟵ X ∪ {elm}
                end
          end
  end
```

analysis. So, for any $k \in \{1, \ldots, K\}$, let $p_k = \underbrace{p_1 \diamond (t,s) \diamond p_1 \diamond \ldots \diamond (t,s) \diamond p_1}_{k\ times}$ be the $k^{\underline{\text{th}}}$ shortest path. It is not difficult to construct such an example. In fact, let us assume that p_1 is an hamiltonean path and $c_{ij} = 0$ for any arc $(i,j) \in \mathcal{A}$ in p_1. Let us also consider that $c_{ij} > K$ holds for any arc $(i,j) \in \mathcal{A}$ such that (i,j) is not an arc of p_1 and the existence of an arc $(t,s) \in \mathcal{A}$ such that $c_{ts} = 1$. In this case, each node is assigned with, at most, K permanent labels. The number of elements that are generated from each permanent label of node i, is exactly the number of arcs whose tail node is i; that is, X has, at most, Km elements. So the required memory space is $\mathcal{O}(Km)$; moreover, using a suitable data structure to represent X, the time complexity is also $\mathcal{O}(Km)$.

To exemplify the algorithm, the problem depicted in figure 1 will be solved for $K = 2$ (see figure 4).

The first two steps are the same as in figure 2 (except for the value of the working variable *count*). Notice that, after step 7 nodes $1''$ and $1'''$ will be picked out from X. That is, $\mathcal{P}_2 = \{p_1, p_2\}$ is determined, where $p_1 = \langle 5,3,4,1 \rangle$ and $p_2 = \langle 5,3,2,4,1 \rangle$; moreover, $count_{t=1} \longleftarrow 2$ and the algorithm finishes.

We must notice that either Dreyfus' algorithm, [11], or **MS** algorithm, [18], and its previous versions, [2, 3, 4, 14], are also label setting algorithms for the shortest paths ranking problem.

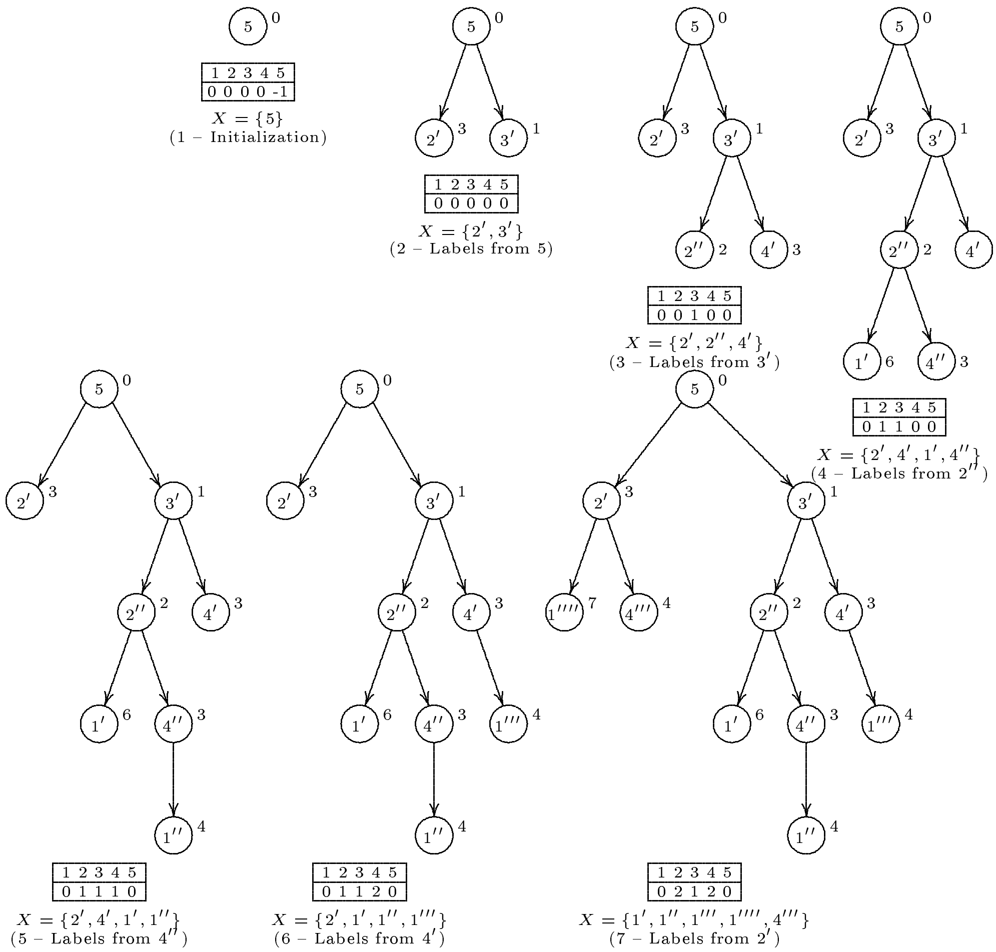


Figure 4: **Seven steps of the Dijkstra's generalization form**

4 Conclusion

Labeling algorithms for ranking of shortest paths were described in this paper. Their theoretical support was also introduced, since they are valued by an Optimality Principle generalization. Some practical considerations were also presented. However, it remains open the practical efficiency of labeling algorithms when compared with deviation ones, [16].

Notice that the ranking shortest loopless paths problem does not hold, in general, the Optimality Principle, [16]. So, labelling algorithms can not be used to solve this constrained ranking paths problem.

Acknowledgments:
The authors acknowledge the referees for their criticism and helpful comments.
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").

References

[1] Y.P. Aneja and K.P.K. Nair. The constrained shortest path problem. *Naval Research Logistics Quarterly*, 25:549–555, 1978.

[2] 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.

[3] J.A. Azevedo, J.J.E.R.S. Madeira, E.Q.V. Martins, and F.M.A. Pires. A shortest paths ranking algorithm, Proceedings of the Annual Conference AIRO'90, Models and Methods for Decision Support, Operational Research Society of Italy, 1001-1011, 1990. (http://www.mat.uc.pt/∼eqvm/cientificos/investigacao/Artigos/K1.ps.gz).

[4] 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.

[5] R.E. Bellman. On a routing problem. *Quarterly Applied Mathematics*, 1:425–447, 1958.

[6] B.V. Cherkassky, A.V. Goldberg, and T. Radzik. Shortest paths algorithms: Theory and experimental evaluation. *Mathematical Programming*, 73:129–196, 1996.

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

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

[9] 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.

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

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

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

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

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

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

[16] E.Q.V. Martins, M.M.B. Pascoal, and J.L.E. Santos. Deviation algorithms for ranking shortest paths. To appear in *The International Journal of Foundations of Computer Science.* (http://www.mat.uc.pt/∼eqvm/cientificos/investigacao/Artigos/deviation.ps.gz).

[17] E.Q.V. Martins, D.M.L.D. Rasteiro, M.M.B. Pascoal, 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).

[18] 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.gz).

[19] E.F. Moore. The shortest path through a maze. Proceedings of the International Symposium on the Theory of Switching, Harvard University Press, 285–292, 1959.

[20] M.M.B. Pascoal. Algoritmos para a enumeração dos K trajectos mais curtos, M. Sc. thesis (in Portuguese); Departamento de Matemática; Universidade de Coimbra, 1997.
(http://www.mat.uc.pt/∼marta/Publicacoes/tese.ps.gz).

[21] 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.

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

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

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