ON THE *K* SHORTEST PATH TREES PROBLEM

ANTONIO SEDEÑO-NODA
Universidad de La Laguna (DEIOC),38271- La Laguna, Tenerife, Spain, asedeno@ull.es

CARLOS GONZÁLEZ-MARTÍN
Universidad de La Laguna (DEIOC),38271- La Laguna, Tenerife, Spain, cgonmar@ull.es

We address the problem for finding the *K* best path trees connecting a source node with any other non-source node in a directed network with arbitrary lengths. The main result in this paper is the proof that the *k*th shortest path tree is adjacent to at least one of the previous (*k*–1) shortest path trees. Consequently, we design an O($f(n,m,C_{\max})+Km$) time and O(*K*+*m*) space algorithm to determine the *K* shortest path trees, in a directed network with *n* nodes, *m* arcs and maximum absolute length $C_{\max}$, where O($f(n,m,C_{\max})$) is the best time needed to solve the shortest simple paths connecting a source node with any other non-source node.

Keywords: Network/Graphs: *K* shortest path trees Problem. Shortest path tree Problem

---

The *shortest path* (SP) problem in a directed network of *n* nodes an *m* arcs with arbitrary lengths on the arcs, finds shortest length paths from a source node to all other nodes or detects a cycle of negative length. The SP problem appears in many important real cases and there are numerous algorithms to solve it (see, for example, Ahuja et al. [1]). The mathematical formulation of the SP problem lets the solutions of the SP problem to be characterized by *path trees*, that is, a tree containing a directed only one path from source node to any non-source node. The determination of the optimal path tree (shortest path tree) can be efficiently determined by Bellman-Ford-Moore (Bellman [3], Ford [7], and Moore [21]) label-correcting algorithm achieving the best strongly polynomial running time of O(*nm*).

In this paper, we consider the *K* shortest path trees problem as the problem to determine the *K* best basis trees (solutions) of the classical mathematical formulation of the SP problem. The determination of the *K* shortest paths in a network has a wide range of applications. Some of them are cited in Eppstein [8]. We are unaware of any previous references to this problem in the literature. A proof is offered which shows that the *k*th best basis tree is adjacent to at least one of the previous *k*–1 best basis trees. In other words, the *k*th best solution is obtained from one of the previous best solutions by exchanging an arc in the basis tree for an arc outside of the basis tree. This results allows an algorithm to be designed running in

$O(Km + f(n,m,C_{\max}))$ time and requiring $O(K+m)$ memory space, where $O(f(n,m,C_{\max}))$ ($> O(m)$) is the best bound to solve the SP problem in a directed network. On the other hand, this problem is similar to the *K* shortest simple paths problem. This last problem has been the subject of a large number of references in the literature. We can chronologically cite Hoffman and Pavley [15], Pollack [23], Yen [25, 26], Lawler [18], Katoh et al. [17] for undirected networks, Perko [22], Brander and Sinclair [4], Martins et al. [20], Hadjiconstantinou and Chirstofides [12], Martins and Pascoal [19] and Hersberger et al. [14]. The best bound to solve the problem in directed networks is reached in the early paper of Yen [25]. Yen's [25] algorithm runs in $O(Kn\, f(n,m,C_{\max}))$. An extended bibliography of several *K* best shortest path problems is available at <`http://www.ics.edu/~eppstein/bibs/kpath.bib`>. We claim that the theoretical results from this paper can be an initial and alternative point of view when considering its application to other *K* best combinatorial optimization problems.

The structure of the paper follows. Section 1 presents the linear programming formulation of the SP problem and the *K* shortest path trees problem are given. In section 2, we introduce the foundations which the algorithm is based on. The adjacency property between the *k*th best solution and some of the previous best solutions is proved. Section 4 contains a detailed pseudo code and an explanation of the proposed algorithm and some procedures. Moreover, the worse case time and space theoretical complexity of the algorithm is proven. Finally, in section 5, we offer our conclusions, lines of future research and open problems.

1. THE SHORTEST PATH TREE AND THE *K* SHORTEST PATH TREES PROBLEMS.

Given a directed network *G* = (*V*, *A*), let V be the set of *n* nodes and let *A* be the set of *m* arcs. The network has a distinguished node *s*, called the *source* node. For each arc $(i,j) \in A$, let $c_{ij} \in \mathbb{R}$ be its length and $C_{\max} = \max_{(i,j)\in A} \{|c_{ij}|\}$. We denote by $\Gamma_i^- = \{ j \in V \mid (j,i) \in A \}$ for all node $i \in V$. The length of a directed path is the sum of the arc lengths in the path. The shortest path tree problem consists in finding a shortest length path from node *s* to every non-source node $i \in V \setminus \{s\}$ or in determining a negative cycle, that is, a directed cycle of negative length.

If a flow x_{ij} is associated with each arc (i, j), a supply $b_s = (n-1)$ with node *s*, and demands $b_i = -1$ for all others nodes $i \neq s$, then the following linear programming problem represents the SP problem (see Ahuja et al. [1]):

$$\text{Minimize} \quad c(x) = \sum_{(i,j)\in A} c_{ij} x_{ij} \qquad (1)$$

subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b_i, \ \forall i \in V \qquad (2)$$

$$x_{ij} \geq 0, \qquad \forall (i, j) \in A \qquad (3)$$

The above problem is a special case of the minimum cost network flow (MCNF) problem. The network simplex algorithm can be used to find the solution to the above problem by taking advantage of the fact that of any basic solution of the MCNF problem is a spanning tree $T \subseteq A$ of *G*. Moreover, every feasible spanning tree *T* is non-degenerate, i.e. $x_{ij} > 0$, $\forall (i, j) \in T$. Therefore, the network simplex algorithm will never perform degenerate pivots for the SP problem. In addition, all feasible spanning trees are trees rooted at node *s* such that the unique path in the tree from node root *s* to every other node is a directed path. We refer to these spanning trees as *directed out-spanning trees*. Note that in this kind of tree, each node $i \in V \setminus \{s\}$ has only one node predecessor in the tree $\left(pred_i(T)\right)$, that is, its in-degree is one. Also, let $T_i^+ = \{ j \in V \mid (i, j) \in T \}$.

Distance labels of the nodes (negative dual variables) corresponding to a basis tree *T* are also obtained by setting $d_s(T) = 0$ and solving $c_{ij} + d_i(T) - d_j(T) = 0$, $\forall (i, j) \in T$. Thus, given a basis tree *T*, we define the reduced cost $\overline{c}_{ij}(T) = c_{ij} + d_i(T) - d_j(T)$, $\forall (i, j) \in A$.

Let *X* be the convex polyhedron defined by constraints (2)-(3) (*decision space*). Two fundamental results in the literature (see, for example, Ahuja et al. [1], Goldfarb et al. [11]) are:

(i) Any feasible solution of the SP problem is a vertex of X and vice-versa.

(ii) Every vertex of X has associated only one directed out-spanning tree.

In the rest of paper, we refer to a directed out-spanning tree as tree (or basis tree). Let $C(T) = \sum_{(i,j)\in T} c_{ij} x_{ij}$ be the value of the function objective associated with the basis tree *T*. We

also define $D_i(T)$ to be the set of descendants of node *i* in the basis tree *T*, that is, the set of nodes in the subtree rooted at *i*, including node *i*. Note that $|D_i(T)| \geq 1$.

In a simplex pivot, an arc $(i, j) \in A \setminus T$ with reduced cost $\bar{c}_{ij}(T)$ and $i \notin D_j(T)$ is added to *T* and $(pred_j(T), j)$ is deleted from *T* yielding a new basis tree T'. The arc flow values are given by $x_{ij} = |D_j(T)|$, $\forall (i, j) \in T$, and the objective function value by $C(T) = \sum_{(i,j)\in T} c_{ij} |D_j(T)|$ or, equivalently, $C(T) = \sum_{i \in V} d_i(T)$. Once a pivot is performed, the distance labels in T' are updated in the following way: $d_k(T') = d_k(T) + \bar{c}_{ij}(T)$, $\forall k \in D_j(T)$. Furthermore, the objective function value is $C(T') = C(T) + \bar{c}_{ij} |D_j(T)|$.

For an optimal tree T^*, we obtain the following optimality conditions: $c_{ij} + d_i(T^*) \geq d_j(T^*)$, $\forall (i, j) \in A$ or, equivalently, $\bar{c}_{ij}(T^*) \geq 0$, $\forall (i, j) \in A$. These inequalities are called *Bellman's optimality conditions*.

The *K* shortest path trees problem consists in determining the *K* best solutions of the problem (1)-(3). In other words, identifying the *K* best basis tree T^k with $k \in \{1, ..., K\}$ such that $C(T^1) \leq C(T^2) \leq ... \leq C(T^K)$ and for any other basis tree $T^p \neq T^k$ with $k \in \{1, ..., K\}$ is $C(T^p) \geq C(T^K)$.

2. FOUNDATIONS.

In this section, we introduce and prove the basic results to the efficient resolution of the *K* shortest path trees problem.

First we need to introduce the following definitions and results which deal with the concept of adjacency between two basis trees:

Definition 1. Two basis tree *T* and T' are adjacent if and only if both have $n-2$ arcs in common, that is, both trees differ in only one arc.

The above definition implies that the basis tree T' can be reached from the basis tree *T* by a simplex pivot where the entering arc is just the arc $(i, j) \in T' \setminus T$ and $(p, q) \in T \setminus T'$ is the leaving arc. Moreover, let *T* and T' be two basis trees that differ in the $p < n$ arcs. Then the following property introduced in Sedeño-Noda and González-Martín [24] holds:

Proposition 1. *If T and T' differ in p < n arcs, where E = $\{(i_1, j_1), \ldots, (i_p, j_p)\}$ are the arcs in T' that are not in T, then:* (1) *E does not contain a directed cycle;* (2) $j_u \neq j_v$ *holds for all* $u, v \in \{1, \ldots, p\}$ *with* $u \neq v$ *;* (3) *These arcs define the smallest simplex pivot sequence to obtain T' from T, and the order in which these simplex pivots are performed is irrelevant.*

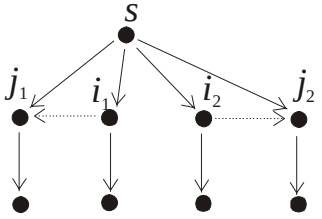
Proposition 2 indicates that to obtain an optimal basis tree T^* from any basis tree *T*, we must choose a set of $p < n$ arcs, $(i_1, j_1), \ldots, (i_p, j_p)$, satisfying $j_u \neq j_v$ for all $u, v \in \{1, \ldots, p\}$ with $u \neq v$. Given any basis tree *T*, we use the term *multiple pivot* for the operation where *p* < *n* arcs are entered simultaneously in *T*, satisfying proposition 2.

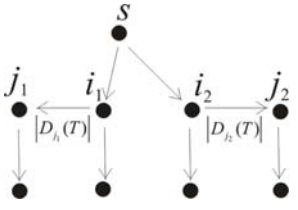
We now need to obtain the best basis tree T', that is, the basis tree with a smaller objective value $C(T') \geq C(T)$, from the basis tree *T* with all its non-tree arcs with non-negative reduced cost (for example an optimal basis tree $T = T^*$). To do so, we must investigate which $p < n$ arcs that satisfy proposition 2 lead to the smallest increase in the objective function of the SP problem when they are introduced into the basis tree *T*, thereby obtaining the basis tree T'. We must investigate the value of objective function when a multiple pivot on basis tree *T* is made in order to identify this set of arcs.

For simplicity consider a multiple pivot with the arcs $\{(i_1, j_1), (i_2, j_2)\}$ with $j_1 \neq j_2$ in tree *T*, obtaining tree T'. Without a loss of generality, we assume that $j_1 \notin D_{j_2}(T)$ (note that if $j_1 \in D_{j_2}(T)$ then $j_2 \notin D_{j_1}(T)$ and we can interchange j_1 by j_2 and vice versa in our arguments). Now, the objective function value of the basis tree T' as a function of $C(T)$ must be considered. Let us examine the following cases:

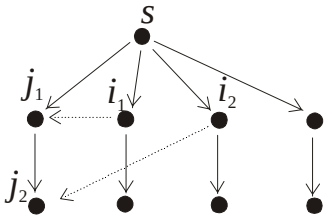
Case A) If $j_1 \notin D_{j_2}(T)$ and $j_1 \notin D_{j_2}(T')$ then, theses subcases must be considered:

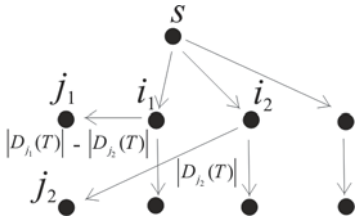
A1). $j_2 \notin D_{j_1}(T)$ and $j_2 \notin D_{j_1}(T') \Rightarrow \quad C(T') = C(T) + \bar{c}_{i_1 j_1}(T)\left|D_{j_1}(T)\right| + \bar{c}_{i_2 j_2}(T)\left|D_{j_2}(T)\right|$



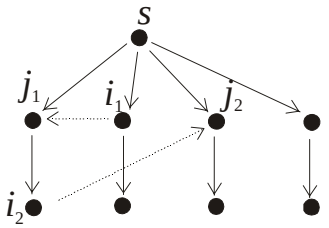


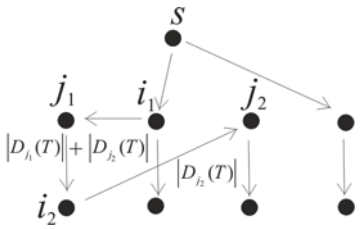
A2). $j_2 \in D_{j_1}(T)$ and $j_2 \notin D_{j_1}(T') \Rightarrow$ $C(T') = C(T) + \overline{c}_{i_1 j_1}(T)\left|D_{j_1}(T)\right| + (\overline{c}_{i_2 j_2}(T) - \overline{c}_{i_1 j_1}(T))\left|D_{j_2}(T)\right|$



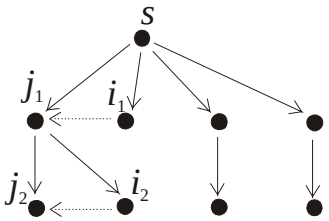


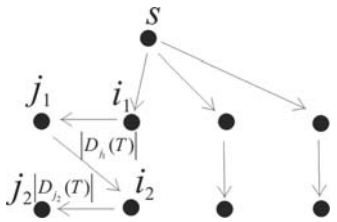
A3). $j_2 \notin D_{j_1}(T)$ and $j_2 \in D_{j_1}(T') \Rightarrow$ $C(T') = C(T) + \overline{c}_{i_1 j_1}(T)\left|D_{j_1}(T)\right| + (\overline{c}_{i_2 j_2}(T) + \overline{c}_{i_1 j_1}(T))\left|D_{j_2}(T)\right|$





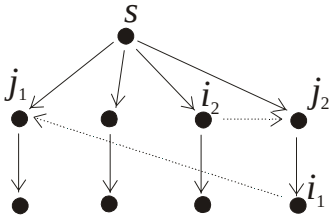
A4). $j_2 \in D_{j_1}(T)$ and $j_2 \in D_{j_1}(T') \Rightarrow$ $C(T') = C(T) + \overline{c}_{i_1 j_1}(T)\left|D_{j_1}(T)\right| + \overline{c}_{i_2 j_2}(T)\left|D_{j_2}(T)\right|$

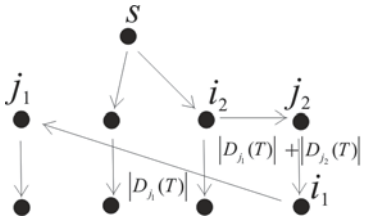




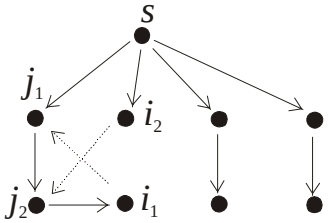
Case B). If $j_1 \notin D_{j_2}(T)$ and $j_1 \in D_{j_2}(T')$ ($j_2 \notin D_{j_1}(T')$) then, these subcases must be considered:

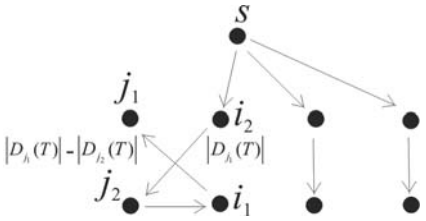
B1). $j_2 \notin D_{j_1}(T)$ and $j_2 \notin D_{j_1}(T') \Rightarrow$ $C(T') = C(T) + \overline{c}_{i_2 j_2}(T)\left|D_{j_2}(T)\right| + (\overline{c}_{i_1 j_1}(T) + \overline{c}_{i_2 j_2}(T))\left|D_{j_1}(T)\right|$





B2). $j_2 \in D_{j_1}(T)$ and $j_2 \notin D_{j_1}(T') \Rightarrow$ $C(T') = C(T) + (\overline{c}_{i_1 j_1}(T) + \overline{c}_{i_2 j_2}(T))\left|D_{j_1}(T)\right| - \overline{c}_{i_1 j_1}(T)\left|D_{j_2}(T)\right|$





From the above cases, we can conclude that:

Lemma 1. *Given a basis tree T, let T' be the basis tree obtained from T making a multiple pivot with two arcs $\{(i_1, j_1),(i_2, j_2)\}$ both with non-negative reduced cost. Then C(T′) is greater than or equal to the objective function value of at least one of the two basis trees that can be obtained by making a simplex pivot with only one arc in the set $\{(i_1, j_1),(i_2, j_2)\}$.*

Proof. Since $D_{j_1}(T) > 0$, $D_{j_2}(T) > 0$, $\overline{c}_{i_1 j_1} \geq 0$ and $\overline{c}_{i_2 j_2} \geq 0$, we have the following results:

- In case A1), we conclude that $\overline{c}_{i_1 j_1}(T)\left|D_{j_1}(T)\right| + \overline{c}_{i_2 j_2}(T)\left|D_{j_2}(T)\right| \geq \overline{c}_{i_1 j_1}(T)\left|D_{j_1}(T)\right|$ and $\overline{c}_{i_1 j_1}(T)\left|D_{j_1}(T)\right| + \overline{c}_{i_2 j_2}(T)\left|D_{j_2}(T)\right| \geq \overline{c}_{i_2 j_2}(T)\left|D_{j_2}(T)\right|$.
- In case A2), since $\left|D_{j_1}(T)\right| > \left|D_{j_2}(T)\right|$, we conclude that $\overline{c}_{i_1 j_1}(T)\left|D_{j_1}(T)\right| + (\overline{c}_{i_2 j_2}(T) - \overline{c}_{i_1 j_1}(T))\left|D_{j_2}(T)\right| \geq \overline{c}_{i_2 j_2}(T)\left|D_{j_2}(T)\right|$. In addition, if $\overline{c}_{i_1 j_1}(T)\left|D_{j_1}(T)\right| + (\overline{c}_{i_2 j_2}(T) - \overline{c}_{i_1 j_1}(T))\left|D_{j_2}(T)\right| < \overline{c}_{i_1 j_1}(T)\left|D_{j_1}(T)\right|$ then, $\overline{c}_{i_1 j_1} > \overline{c}_{i_2 j_2}$ and vice versa.
- In case A3), we conclude that $\overline{c}_{i_1 j_1}(T)\left|D_{j_1}(T)\right| + (\overline{c}_{i_2 j_2}(T) + \overline{c}_{i_1 j_1}(T))\left|D_{j_2}(T)\right| \geq \overline{c}_{i_2 j_2}(T)\left|D_{j_2}(T)\right|$ and also that $\overline{c}_{i_1 j_1}(T)\left|D_{j_1}(T)\right| + (\overline{c}_{i_2 j_2}(T) + \overline{c}_{i_1 j_1}(T))\left|D_{j_2}(T)\right| \geq \overline{c}_{i_1 j_1}(T)\left|D_{j_1}(T)\right|$.
- In case A4) we obtain the same conclusions as in case A1).
- In case B1), we obtain the same conclusions as in case A3).
- In case B2), we have a special situation. A basis tree has not been obtained by making a pivot only with the arc (i_1, j_1) because $i_1 \in D_{j_1}(T)$. Thus, since $\left|D_{j_1}(T)\right| > \left|D_{j_2}(T)\right|$, we obtain that $(\overline{c}_{i_1 j_1}(T) + \overline{c}_{i_2 j_2}(T))\left|D_{j_1}(T)\right| - \overline{c}_{i_1 j_1}(T)\left|D_{j_2}(T)\right| \geq \overline{c}_{i_2 j_2}(T))\left|D_{j_2}(T)\right|$.

From the above comparisons, we arrive to the conclusion of the lemma. □

Given a basis tree *T*, we denote by $A(T)$ a subset of arcs of $A \backslash T$ and $\Gamma_i^+(A(T)) = \{ j \in V \mid (i, j) \in A(T) \}$ (the set of nodes adjacent to node *i* in the directed graph (*V*, $A(T)$)). Then we define $(i, j)_{A(T)} = \arg \min_{(u,l) \in A(T)} \{ \overline{c}_{ul}(T)\left|D_l(T)\right| : u \notin D_l(T) \}$. We do not consider in the previous minimum the arcs (u, l) such that $u \in D_l(T)$ because a simple simplex pivot with any of these arcs does not allow us to obtain a basis tree. Then, we obtain the next result.

Lemma 2. *Given a basis tree T with a set of arcs* $A(T)$ *with non-negative reduced cost. Let* T' *be the basis tree obtained from T making a simplex pivot with the arc* $(i,j)_{A(T)} \in A(T)$. *Then any arc in the set* $A(T') = A(T) - \{(i,j)_{A(T)}\}$ *has a non-negative reduced cost with respect to the basis tree* T'.

Proof. Given a basis tree *T*, when a simplex pivot is made with the entering arc $(i,j) = (i,j)_{A(T)} \in A(T)$,the basis tree T' is obtained, and only the distance labels of the nodes in $D_j(T)$ increase by $\bar{c}_{i,j}(T)$.

First, we must consider the next cases to verify the sign of each arc (u,l) in $A(T') = A(T) - \{(i,j)\}$ with $u \notin D_l(T)$:

Case 1) if $u \in D_j(T)$ and $l \in D_j(T)$ or $u \notin D_j(T)$ and $l \notin D_j(T)$ then, we have $d_u(T') - d_l(T') = d_u(T) - d_l(T)$ and therefore $\bar{c}_{ul}(T') = \bar{c}_{ul}(T) \geq 0$.

Case 2) if $u \in D_j(T)$ and $l \notin D_j(T)$ then

$$\bar{c}_{ul}(T') = c_{ul} + d_u(T') - d_l(T') = c_{ul} + d_u(T) + \bar{c}_{ij}(T) - d_l(T) = \bar{c}_{ul}(T) + \bar{c}_{ij}(T) \geq 0.$$

Case 3) if $u \notin D_j(T)$ and $l \in D_j(T)$ then

$\bar{c}_{ul}(T') = c_{ul} + d_u(T') - d_l(T') = c_{ul} + d_u(T) - d_l(T) - \bar{c}_{ij}(T) = \bar{c}_{ul}(T) - \bar{c}_{ij}(T)$, but since $(i,j)_{A(T)} = \arg\min_{(u,l)\in A(T)} \{\bar{c}_{ul}(T)|D_l(T)| : u \notin D_l(T)\}$ and $|D_j(T)| > |D_l(T)|$ then, $\bar{c}_{ij}(T)|D_j(T)| \leq \bar{c}_{ul}(T)|D_l(T)| \leq \bar{c}_{ul}(T)|D_j(T)|$ and therefore, $\bar{c}_{ul}(T) - \bar{c}_{ij}(T) \geq 0$.

Now, we consider an arc (u,l) in $A(T') = A(T) - \{(i,j)\}$ with $u \in D_l(T)$. Then, only the previous cases 1 and 2 must be considered, since case 3 is not possible, because if $l \in D_j(T)$ and $u \in D_l(T)$, then $u \in D_j(T)$. Therefore, using similar arguments it is proved that $\bar{c}_{ul}(T') \geq 0$.

Therefore, in all of the above cases $\bar{c}_{ul}(T') \geq 0$ and, any arc in $A(T')$ has a non-negative reduced cost. □

Note that when a simplex pivot with the entering arc $(i, j)_{A(T)}$ is made, only the leaving arc $(pred_j(T), j)$ could have negative reduced cost for the basis tree T', but this arc does not belong to $A(T')$.

From the previous results, we can establish the following:

Lemma 3. *Given a basis tree T, let T' be the basis tree obtained from T making a multiple pivot with the arcs $\{(i_1, j_1),(i_2, j_2),...,(i_p, j_p)\}$ with non-negative reduced cost and with $2 \le p < n$. Then $C(T')$ is greater than or equal to the objective function value of at least one of the p basis trees that can be obtained by making a simplex pivot with only one arc in the set $\{(i_1, j_1),(i_2, j_2),...,(i_p, j_p)\}$.*

Proof. Let $A(T) = \{(i_1, j_1),(i_2, j_2),...,(i_p, j_p)\}$. Note that any arc in $A(T)$ a has non-negative reduced cost for the basis tree *T*. From Lemma 1, we have already proven the statement of this lemma for $p = 2$. Now, we assume that $p = 3$ and determine $(i, j)_{A(T)}$. Thus, the multiple pivot with the arcs $A(T)$ can be considered in the next order: first we make a simplex pivot with the arc $(i, j)_{A(T)}$ obtaining the basis tree T' and then a multiple pivot with the two remainder arcs in $A(T') = A(T) - \{(i, j)_{A(T)}\}$. Note that from lemma 2 any arc in $A(T')$ has a non–negative reduced cost for the basis tree T'. Therefore, the basis tree T' and the set $A(T')$ satisfy the conditions of lemma 1. That is, the basis tree obtained making multiple pivots with the two arcs in $A(T')$ has an objective function value greater than or equal to at least one of the two basis trees that can be obtained by making a simple simplex pivot with only one arc in the set $A(T')$ in T'. Thus, consider the basis tree T'' obtained by making a simple simplex pivot with any arc $(u,l) \in A(T')$. Note that T'' can be obtained from *T* making a multiple pivot with the arcs $(i, j)_{A(T)}$ and (u,l). In other words, the set of arcs $(i, j)_{A(T)}$ and (u,l) for the basis tree *T* satisfy the conditions of lemma 1. Therefore, a basis tree with the lowest increase in the objective function value $C(T)$ can be obtained from *T* by making a simple simplex pivot with the arc $(i, j)_{A(T)}$. Thus, we have proved the statement of this lemma for $p = 3$. By induction and using the above arguments, the statement of this lemma is proven for any $p < n$. □

We are interested in generating the *K* best shortest simple path trees in order without repeating the calculation of the same best solution. To do so, given a basis tree *T*, the entering arc $(i,j)=(i,j)_{A(T)}=\arg\min_{(u,l)\in A(T)}\left\{\bar{c}_{ul}(T)\middle|D_l(T)\right|:u\notin D_l(T)\}$ and the basis tree T' obtained by making a simplex pivot with the entering arc (i,j), we modify the respective set of non-tree arcs as follows: $A(T)=A(T)-\{(i,j)\}$ and $A(T')=A(T)-\{(u,l)\in A(T):l=j\}$. Note that since $A(T')$ does not contain any incoming arc in node *j*, any basis tree obtained from T' subsequently contains the arc (i,j). In addition, any basis tree obtained from *T* does not contain the arc (i,j) (binary partition strategy). From these comments, it is clear that the determination of the same basis tree is not performed twice or more times.

Finally, based on lemmas 2 and 3; we come to the first conclusion included in the main results in this paper:

Theorem 1. *The kth best shortest path tree is adjacent to at least one of the previous* (*k*–1)*th best shortest path trees*.

Proof. Let $T^1=T^*$ be the first best basis tree, that is, an optimal shortest simple path tree. Clearly, any arc in $A(T^1)=A\setminus T^1$ has a non-negative reduced cost and *G* does not have negative length cycle. Otherwise T^1 is not an optimal shortest path tree. From Lemma 3, we obtain a second best basis tree T^2 by making a simple simplex pivot in T^1 with the arc $(i,j)_{A(T^1)}$. Clearly, T^2 is adjacent to basis tree T^1. Now, let $A(T^1)=A(T^1)-\left\{(i,j)_{A(T^1)}\right\}$ and $A(T^2)=A(T^1)-\{(u,l)\in A(T^1):l=j\}$ be the associated non-tree arcs sets of the trees T^1 and T^2, respectively. From lemma 2, any arc in $A(T^2)\subset A(T^1)$ has a non-negative reduced cost with respect to T^2. Therefore, let $T^{1'}$ be the basis tree obtained from T^1 making a simple simplex pivot with the entering arc $(i,j)_{A(T^1)}$. Similarly, let be $T^{2'}$ be the basis tree obtained from T^2 by making a simplex pivot with the entering arc $(i,j)_{A(T^2)}$. From Lemma 3, $T^{1'}$ and $T^{2'}$ are the closest basis trees that can be obtained from T^1 and T^2, respectively, using their corresponding set of arcs $A(T^1)$ and $A(T^2)$. Clearly, the third best basis tree is $T^3=\arg\min\left\{C(T)\,|\,T\in\left\{T^{1'},T^{2'}\right\}\right\}$. If T^3 equals $T^{1'}$ then T^3 is adjacent to T^1 and we set

$A(T^1) = A(T^1) - \left\{(i,j)_{A(T^1)}\right\}$ and $A(T^3) = A(T^1) - \left\{(u,l) \in A(T^1) : l = j\right\}$. Otherwise, if T^3 equals $T^{2'}$ then T^3 is adjacent to T^2 and we set $A(T^2) = A(T^2) - \left\{(i,j)_{A(T^2)}\right\}$ and $A(T^3) = A(T^2) - \left\{(u,l) \in A(T^2) : l = j\right\}$. Note again that from Lemma 2, any arc in the resulting set of non-tree arcs has a non-negative reduced cost and, therefore the conditions of Lemma 3 are satisfied. Therefore, by induction it is proved that $T^k = \arg\min\left\{C(T) \mid T \in \left\{T^{1'},...,T^{k'}\right\}\right\}$ where $T^{p'}$ is the basis tree obtained from the *pth* best basis tree T^p by making a simplex pivot with the entering arc $(i,j)_{A(T^p)} = \arg\min_{(u,l)\in A(T^p)}\left\{\bar{c}_{ul}(T^p)\left|D_l(T^p)\right| : u \notin D_l(T^p)\right\}$. In other words, the theorem holds. □

3. AN EFFICIENT ALGORITHM FOR THE *K* SHORTEST PATH TREES PROBLEM.

This section introduces an algorithm to solve efficiently the *K* shortest path trees problem. We begin introducing additional notation.

Given a basis tree *T*, the proposed method uses distance labels $d_u(T)$ and the predecessor labels $pred_u(T)$ for all $u \in V$. The algorithm also needs to know the value of $\left|D_u(T)\right|$ for all $u \in V$ and the objective value $C(T)$. Thus, given a basis tree *T*, we assume that in the adjacency node list $T_i^+ = \left\{j \in V \mid (i,j) \in T\right\}$ the values of c_{ij} are stored. We initially set $C(T) = 0$, $d_s(T) = 0$, $pred_s(T) = s$ and $\left|D_u(T)\right| = 0$, $\forall u \in V$. Then, the *ComputingLabels* procedure is called with $u = s$ for a given basis tree *T* to calculate the previous information in O(*n*) time.

Procedure (CL) ComputingLabels(u, var $C(T)$, var $d(T)$, var $pred(T)$, var $\left|D(T)\right|$, T);

(1) **For** all $l \in T_u^+$ **do**
(2) $pred_l(T) = u$;
(3) $d_l(T) = d_u(T) + c_{ul}$;
(4) ComputingLabels($l, C(T), d(T), pred(T), \left|D(T)\right|, T$);
(5) $\left|D_u(T)\right| = \left|D_u(T)\right| + \left|D_l(T)\right|$;
(6) $\left|D_u(T)\right| = \left|D_u(T)\right| + 1$;
(7) $C(T) = C(T) + d_u(T)$;

Additionally, we associated a subset of non-tree arcs $A(T)$ with each basis tree *T*. For each calculated basis tree T^p, the arc $(i,j)_{A(T^p)} = \arg\min_{(u,l)\in A(T^p)} \left\{ \overline{c}_{ul}(T^p) \left| D_l(T^p) \right| : u \notin D_l(T^p) \right\}$ is calculated and it is stored together with the index *p* indicating the associated *p*th best basis tree in a heap using as key the value $\overline{c}_{ij}(T^p)\left|D_j(T^p)\right| + C(T^p)$. We denote this heap by *H* in the algorithm. Assuming that *t* is the size of a heap, the operation *Insert* requires an effort O(log *t*) and the operation of extracting the element of the min-key (*Extract First*) takes O(1) time. The *Create Heap* operation takes O(1) time.

In order to simplify the examination of the set of arcs $A(T)$ for a given basis tree *T*, we maintain an additional Boolean label named $eligible_i(T)$ for each node $i \in V$. $eligible_i(T)$ is FALSE if and only if the arc $(pred_i(T), i)$ can not be chosen to leave the basis tree *T* (equivalently, no arc arriving at node *i* can be selected to enter into the basis tree *T*). Otherwise, $eligible_i(T)$ is TRUE. On the other hand, given a basis tree *T* and its corresponding set of non-tree arcs $A(T)$, we assume that in the adjacency node list $\Gamma_i^+(A(T)) = \{ j \in V \mid (i,j) \in A(T) \}$ the value of c_{ij} and Boolean label $Arc\_eligible_{ij}(T)$ are stored. $Arc\_eligible_{ij}(T)$ is TRUE for arc $(i,j) \in A(T)$ if and only if this arc can be chosen to enter into the basis tree *T*. Initially the label $Arc\_eligible_{ij}(T)$ is TRUE for all arc (i,j) in *G*.

Using the above notation, one way to easily implement the selection of the arc $(i,j) = (i,j)_{A(T)} = \arg\min_{(u,l)\in A(T)} \left\{ \overline{c}_{ul}(T) \left| D_l(T) \right| : u \notin D_l(T) \right\}$ consist in applying the following recursive procedure:

Procedure (SMA) Searching_Minimum_Arc(u, var C, $T, A(T), d(T), \left|D(T)\right|, eligible(T)$);

(1) $visited_u$ = TRUE ;
(2) **For** all $l \in T_u^+$ **do**
(3) Searching_Minimum_Arc (l, C);
(4) **For** all $l \in \Gamma_u^+(A(T))$ **do**
(5) **If** ($visited_l$ == FALSE) **and** ($eligible_l(T) == TRUE$) **and**
($Arc\_eligible_{ul}(T)$ == TRUE) **and** ($(c_{ul} + d_u(T) - d_l(T)) \cdot \left|D_l(T)\right| < C$) **Then**
(6) $C = (c_{ul} + d_u(T) - d_l(T)) \cdot \left|D_l(T)\right|$;
(7) $i = u$;
(8) $j = l$;
(9) $visited_u$ = FALSE ;

The procedure SMA is called with parameters $u = s$ and the variable $C = \infty$ to determine the arc $(i, j)_{A(T)} = \arg \min_{(u,l)\in A(T)} \left\{ \overline{c}_{ul}(T) \middle| D_l(T) \middle| : u \notin D_l(T) \right\}$ for a given basis tree *T* with the set of non-tree arcs $A(T)$. Moreover, when a non-tree arc (u,l) is examined in the procedure and $visited_l == \text{TRUE}$, this means that $u \in D_l(T)$ and, therefore, this arc is not considered for selection. The only local variable in the procedure is l. All elements used in the procedure are defined out of this procedure (initially all nodes are not visited). This procedure returns a pointer to the arc (i, j) belonging to $A(T)$ if it exist, otherwise it returns NULL. As *T* is a basis tree, each node $u \in V$ is reached exactly once when this procedure is called from node *s*. For each node, the set of arcs (u,l) with $l \in \Gamma_u^+(A(T))$ are scanned. Thus the computational effort performed by this procedure is O(*m*).

Finally, we use an additional data structures as in Gabow [9] to reduce the memory space needed in the algorithm. Thus, let us assume that the first *k* basis trees $T^{k'}$, $k' \in \{1,...,k\}$, have been calculated. Then, we use the following structures to store these basis trees as a *directed out tree*: $father[k']$ stores the index *p* associated with basis tree T^p and a pointer to the entering arc (i, j) in *G* that leads to obtaining a basis tree $T^{k'}$ ($father[k'] = \{p, (i, j)\}$). Each element of the list $sons[k']$ stores the index and a pointer to the entering arc (i, j) in *G* that allowed the basis tree T^p be obtained from $T^{k'}$. The list $sons[k']$ is arranged in such a way that the indices increase from left to right ($sons[k'] = \{\{ p_1, (i, j)_1 \},\ldots,\{ p_r, (i, j)_r \}\}$ and $p_1 < ... < p_r$).

In Gabow [9] the following sets are described to build T^k and partially build $A(T^k)$: Let B be the set of basis trees in the path from T^1 to T^k in the tree of trees, then define:

$O_1 = \left\{ \text{the entering arc } (i, j)_p \text{ to obtain } T^p \middle| \, T^p \text{ is in } B \text{ and } p > 1 \right\}$

$O_2 = \left\{ \text{the leaving arc } (pred_j(T), j)_p \text{ to obtain } T^p \middle| \, T^p \text{ is in } B \text{ and } p > 1 \right\}$

$T^k = T^* \cup O_1 - O_2$

$I = \left\{ \text{the entering arc } (i, j)_p \text{ to obtain } T^p \middle| \, T^p \text{ is the left brother of } T^l \text{ in } B \text{ for all } T^l \text{ in } B \text{ and } l > 1 \right\}$

$I_k = \left\{ \text{the entering arc } (i, j)_p \text{ to obtain } T^p \middle| \, T^p \text{ is a son of } T^k \right\}$

We additionally define the set

$J_k = \left\{ (l, j) \in A \setminus T^* \middle| \, (i, j)_p \text{ is the entering arc to obtain } T^p \text{ belonging to } B \text{ for } p > 1 \right\}$

Then $A(T^k) = A(T^*) - J_k - I - I_k$. Note that $O_1 \cup O_2 \subseteq J_k$.

Using this information, any basis tree T^k and its corresponding set of non-tree arcs $A(T^k)$ can be derived from the initial optimal basis tree T^* and $A(T^*)$ by the next two procedures.

Procedure (BTA) BuildingT&A(T)(k, $father$, $sons$ var $pred(T)$, var $eligible(T)$, var T, var $A(T)$);

(1) **If** $(k \neq 1)$ **then**
(2) **For** all $\{l,(i,j)\} \in sons[father[k].p]$ with $l < k$ **do**
(3) $Arc\_eligible_{ij}(T) = \text{FALSE}$;
(4) BuildingT&A(T)($father[k].p$, $father$, $sons$, $pred(T)$, $eligible(T)$, T, $A(T)$);
(5) $T = T \cup \{father[k].(i,j)\} \setminus \{(pred_j(T), j)\}$
(6) $pred_j(T) = i$;
(7) $eligible_j(T) = FALSE$;

The previous procedure is called by the next procedure:

Procedure (CBTA) CallingBTA(k, $father$, $sons$, var $pred(T)$, var $eligible(T)$, var T, var $A(T)$);

(1) $Arc\_eligible_{ij}(T) = \text{TRUE} \quad \forall (i,j) \in A(T)$;
(2) **For** all $\{l,(i,j)\} \in sons[k]$ **do**
(3) $Arc\_eligible_{ij}(T) == \text{FALSE}$;
(4) BuildingT&A(T)(k, $father$, $sons$, $pred(T)$, $eligible(T)$, T, $A(T)$);

The procedure CBTA is called with $T = T^*$ and $pred_u(T) = pred_u(T^*)$ for all $u \in V$, where T^* is an optimal tree and the index *p* of the basis tree to be constructed. The output of this procedure is $T = T^k$ and $A(T)$ with some arcs marked as ineligible. Line (1) in procedure CBTA initializes the flags $Arc\_eligible_{ij}(T) = TRUE$ for each arc $(i, j) \in A(T)$. Furthermore, in lines (2)-(3) of procedure CBTA, we set $Arc\_eligible_{ij}(T)$ equal to FALSE for all $\{p', (i, j)\} \in sons[k]$, that is, each arc in $A(T^k)$ that was selected is made ineligible ($A(T^k) = A(T^*) - I_k$). Line (4) calls the procedure BTA.

Lines (4)-(6) in procedure BTA lead to recursively building the basis tree $T^{k'}$, that is, these lines backtracking on index k using the *father* labels until $k = 1$. This process lets us identify the needed exchanges in T^* to obtain *T* and we need line (6) to correctly compute $T^{k'}$

($T^k = T^* \cup O_1 - O_2$). On the other hand, lines (2), (3) and (7) lets us determine which additional arcs in $A(T)$ are not eligible to enter in the basis tree T. Recall that when we make a simplex pivot with arc $(i, j) \in A(T)$ in T then, we set $A(T) = A(T) - \{(i, j)\}$ and $A(T') = A(T) - \{(u, l) \in A(T) : l = j\}$. In other words, we set $Arc\_eligible_{ij}(T)$ and $eligible_j(T')$ equal to FALSE. Thus, if $T^{k'}$ is obtained from T^* by a sequence of simplex pivots (i_w, j_w) with $w \in \{1,...,u\}$ then, $eligible_w(T^{k'})$ is FALSE $\forall w \in \{1,...,u\}$. Additionally in lines (2)-(3), if $T^{k'}$ is a son of the basis tree with index $father[k'].p$, then, we set $Arc\_eligible_{ij}(T^{k'})$ equal to FALSE $\forall \{p', (i, j)\} \in sons[\, father[k'].p]$ such that $p' < k'$ and then, we set $k = father[k'].p$ and the process is repeated until $k = 1$ ($A(T^k) = A(T^k) - J_k - I$).

Note that Line (1) of the procedure CBTA requires an effort O(m) and line (4) is the called to the procedure BTA. The recursive procedure BTA is called at most $n-1$ times, since the depth of the tree of trees is at most $n-1$. Line (3) of this procedure and line (3) of the procedure CBTA are executed at most m times in overall. Since for any basis tree $T^{k'}$ in the tree of trees, the number of its sons ($|I_{k'}|$) plus the number of its ancestors belonging to the set I ($|I|$) is at most m. Therefore the computational effort of procedure CBTA and BTA is O(n+m).

Taking into account the above notation and remarks, we introduce the algorithm to solve the problem.

K Shortest Path Trees (KSPT) **Algorithm**;

/* Initialitation */

(1) Let T^* be an optimal basis tree; Store $A(T) = A \setminus T^*$ with $Arc\_eligible_{(i,j)} = \text{TRUE}$ $\forall (i,j) \in A(T)$;

(2) Set $k = 1$; $T = T^*$; $eligible_u(T) = TRUE$ $pred_u(T) = pred_u(T^*)$ $\forall u \in V$;

(3) Set $C(T) = 0$; $d_s(T) = 0$; $|D_u(T)| = 0$ $\forall u \in V$;

(4) CL$(s, C(T), d(T), pred(T), |D(T)|, T)$;

(5) Create Heap H;

(6) $visited_u = FALSE$ $\forall u \in V$;

(7) Let (i,j) determined by SMA$(s,\ C = \infty\ T, A(T), d(T), |D(T)|, eligible(T))$;

(8) **If** $((i,j) \neq$ NULL) **then** Insert $\{(i,j),\ k,\ C + C(T)\}$ in H;

/* loop */

(9) **While** $((k < K)$ **and** $(H \neq \varnothing))$

(10) Extract first $\{(i,j), C, p\}$ of H;

(11) $father[k+1] = \{p, (i,j)\}$; Add $\{k+1, (i,j)\}$ at the end of $sons[p]$;

(12) $T = T^*$; $eligible_u(T) = TRUE$ $pred_u(T) = pred_u(T^*)$ $\forall u \in V$;

(13) CBTA$(p, father, sons, pred(T), eligible(T),\ T,\ A(T))$

(14) Set $C(T) = 0$; $d_s(T) = 0$; $|D_u(T)| = 0$ $\forall u \in V$;

(15) CL$(s, C(T), d(T), pred(T), |D(T)|, T)$

(16) Let (i,j) determined by SMA$(s,\ C = \infty\ T, A(T), d(T), |D(T)|, eligible(T))$;

(17) **If** $((i,j) \neq$ NULL) **then** Insert $\{(i,j),\ p,\ C + C(T)\}$ in H;

(18) Set $k = k + 1$;

(19) $T = T \cup \{father[k].(i,j)\} \setminus \{(pred_j(T), j)\}$

(20) $eligible_j(T) = FALSE$;

(21) Set $C(T) = 0$; $d_s(T) = 0$; $|D_u(T)| = 0$ $\forall u \in V$;

(22) CL$(s, C(T), d(T), pred(T), |D(T)|, T)$;

(23) Let (i,j) determined by SMA$(s,\ C = \infty\ T, A(T), d(T), |D(T)|, eligible(T))$;

(24) **If** $((i,j) \neq$ NULL) **then** Insert $\{(i,j),\ k,\ C + C(T)\}$ in H;

/* end of the loop */

The algorithm starts with an optimal basis tree T^* that is stored as the first best basis tree and its corresponding set of arcs $A(T) = A \setminus T^*$ is also stored. The flags *Arc_eligible* for each arc in $A(T)$ is set to TRUE. The index of the number of best solutions determined k is set to 1. Then the procedure CL is called to compute all labels associated with basis tree T. The arc (i, j) is determined by calling the procedure *SMA*$(s,\ C = \infty,\ T,\ A(T),\ d(T),\ |D(T)|,$

eligible(T)). The heap H is created and the element $\{(i,j)$, 1, $\bar{c}_{ij}(T)\cdot\left|D_j(T)\right|+C(T)\}$ is inserted in *H* wherever arc (i, j) exists. Then, the algorithm starts with a loop until the *K* best solutions are identified or no more feasible solutions are possible. Thus, in any iteration in the algorithm, the first element in the heap is extracted. This element identifies the way to obtain the (k +1)th best solution. In the algorithm $father[k+1]=\{p,(i,j)\}$ lets us determine T^{k+1} . Moreover, adding $\{k+1,(i,j)\}$ at the end of $sons[p]$ will let us correctly identify the set $A(T)$ for T^{k+1} and for all descendants of T^{k+1} in the tree of trees. Furthermore, the construction of the set of arcs $A(T)$ avoids determining each basis tree more than once, as previously mentioned. Now, in the algorithm the basis tree T^p and $A(T^p)$ are reconstructed calling procedure CBTA and the labels of the basis tree T^p are calculated by calling procedure CL. Then, the new $(i,j)_{A(T^p)}$ arc is found by calling the procedure *SMA* for the basis tree T^p. The resulting arc (if it exists) and the index *p* are stored in the heap *H* using the key value $C+C(T^p)$ where *C* is the value calculated in procedure SMA. Next in the algorithm, the index *k* is increased and the basis tree T^k and its associated set of non-tree arcs $A(T^k)$ are built from the basis tree T^p and the set $A(T^p)$ (lines (19)-(20)). The necessary labels of the tree T^k are calculated by calling procedure CL. Finally, for this new best basis tree, the arc $(i,j)_{A(T^k)}$ is determined and the element $\{(i,j)=(i,j)_{A(T^k)}$, *k*, $\bar{c}_{ij}(T^k)\cdot\left|D_j(T^k)\right|+C(T^k)\}$ is inserted in *H*.

Theorem 2. *The KSPT algorithm computes the K shortest path trees in O(Km+* $f(n,m,C_{\max})$*) time and O(K+m) space in directed graph G.*

Proof. In the beginning of the algorithm, the determination of $T^1=T^*$ requires O($f(n,m,C_{\max})$) time, that is O(min{$nm,\sqrt{n}m\log(nC)$}) (see Bellman [3], Ford [7], Moore [21] for example for the first bound and Goldberg [10] for the second bound) for a network with possible negative arc lengths or O(min{ $m+n\log n, m\log\log C, m+n\sqrt{\log C}$ }) for a network with non-negative arc lengths (se Ahuja et al. [1] to find the references of these bounds of the modified Dijkstra [6] algorithm). Storing $A(T)=A\setminus T^*$ and making all arc eligible needs O(*m*) time. Lines (2)-(3), (6) involve an O(*n*) time. The procedure CL in line

(4) involves O(n) time. The calculation of the arc (i, j) by the procedure *SMA* requires an effort O(m) and the operation of create (line (5)) and insert (line (8)) in the heap takes O(1) time. Clearly, the algorithm makes at most K iterations. In each iteration of the algorithm, the procedure *CBTA* is called once and the procedures *CL*, *SMA* are called twice requiring O(n+m) time overall and two insert heap operations are made in O(log k +log (k+1)). The operations relative to lines (10)-(11), (18)-(20) are made in O(1) time. Thus, the worst case complexity of the algorithm is O($f(n,m,C_{\max}) + \sum_{k=1}^{K-1}(\log k + \log(k+1) + n + m)$) = O($f(n,m,C_{\max})$ +Km + Klog$K$) time and, since $K < 2^m$, then O($f(n,m,C_{\max})$ +Km) time. On the other hand, the space required by the algorithm is O(K + m), since the *father*, *sons* and *heap* structures require O(K) space; the basis tree T and its corresponding labels need O(n) space and the storing $A(T)$ employs O(m) space. □

4. CONCLUSIONS.

From this paper, we conclude that the K shortest path trees problem has the same difficulty as the K minimum spanning trees problem (see Katoh et al. [16]). This result is possible since in both problems, the kth best solution is adjacent to at least one of the k–1 best previous solutions. Thus, we design a similar algorithm that takes the advantage of the pivot (exchange) operation for the basis tree of the shortest path tree problem formulated by constraints (1)-(3). Furthermore, since problem (1)-(3) is a particular case of the K best minimum cost flow problem, an open problem consists in verifying if the main result of this paper holds for this problem. An early result for the K best minimum cost flow problem was derived by Hamacher [13]. Moreover, since the shortest simple path between two pair of nodes is a particular case of problem (1)-(3), we ask us self if it is possible to modify the proposed algorithm to obtain an efficient algorithm to solve the K shortest simple paths problem? If the answer is affirmative, the bound of Yen [25] will be significantly improved. On the other hand, the results addressed in this paper are fundamental to develop new algorithms for the multiobjective shortest path problem from one source node to all other non-source nodes in a network (see Azvedo et al. [2] and Climaco and Martins [5]).

ACKNOWLEDGMENTS

This work has been partially supported by Spanish Government Research Project MTM2006-10170.

REFERENCES

[1] Ahuja, R., T. Magnanti, J. B. Orlin. 1993. *Network Flows*. Prentice-Hall, inc.

[2] Azvedo, J., E.Q.V. Martins. 1991. An algorithm for the multiobjective shortest path problem on Acyclic networks. *Investigacao Operacional* **11** 52-69.

[3] Bellman, R. 1958. On a Route Problem. *Quart. Of Appl. Math.* **16** 87-90.

[4] Brander, A., M. Sinclair (1995). A comparative study of *K*-shortest path algorithms. *In Proc. Of 11th UK Performance Engineering Workshop* 370-379.

[5] Climaco, J.C.N., E.Q.V. Martins. 1982. A bicriterion shortest path algorithm. *European Journal of Operational Research* **11** 399-404.

[6] Dijkstra E. W. 1959. A note on two problems in connection with graphs. *Numer. Math.* **1** 269-271.

[7] Ford, L. R. 1956. *Network Flow Theory*. The Rand Corporation Report P-923, Santa Monica, Calif.

[8] Eppstein, D. 1999. Finding the *K* shortest paths. *Siam Journal on Computing* **28** 653-674.

[9] Gabow, H. N. 1977. Two Algorithms for Generating Weighted Spanning Trees in Order. *Siam Journal on Computing* **6** (1) 139-150.

[10] Goldberg, A. V. 1995. Scaling Algorithms for the Shortest Paths Problem. *Siam Journal on Computing* **24** 494-504.

[11] Goldfarb D., J. Hao, S. R. Kai. 1990. Efficient Shortest Path Simplex Algorithms. *Operations Research* **38** 624-628.

[12] Hadjiconstantinou, E., N. Chirstofides. (1999). An efficient implementation of an algorithm for finding *K* shortest simple paths. *Networks* **34** 88-101.

[13] Hamacher, H. W. 1995. A note on *K* best network flows. *Annals of Operations Research* **57** 65-72.

[14] Hershberger, J., M. Maxel, S. Suri. (2003). Finding the *k* Shortest Simple Paths: a new algorithm and its implementation. *Proc. 5th Worksh. Algorithm Engineering & Experiments (ALENEX), SIAM. To appear in ACM transactions on Algorithms* (2007).

[15] Hoffman, W., R. Pavley. 1959. A method for the solution of the *N*th best path problem. *Journal of the ACM* **6** 506-514.

[16] Katoh, N., T. Ibaraki, H. Mine. (1981). An Algorithm for finding *K* Minimum Spanning Trees. *Siam Journal on Computing* **10** 247-255.

[17] Katoh, N., T. Ibaraki, H. Mine. (1982). An efficient Algorithm for *K* Shortest Simple Paths. *Networks* **12** 411-427.

[18] Lawler, E. L. (1972). A procedure for computing the *K* best solutions to discrete optimization problems and its application to the shortest path problem. *Magnagement Science* **18** 401-405.

[19] Martins, E.Q.V., M. M. B. Pascoal. 2000. A new implementation of Yen's ranking loopless paths algorithm. *Submitted for publication. Universidade de Coimbra, Portugal*.

[20] Martins, E.Q.V., M. M. B. Pascoal, J. Santos. 1997. A new algorithm for ranking loopless paths algorithm. *Technical report, Universidade de Coimbra, Portugal*.

[21] Moore, Z. F. (1957). The Shortest Path Through a Maze. *In Proceedings of the International Symposium on Theory of Switching, Part II* 285-292.

[22] Perko, A. (1986). Implementation of algorithms for K shortest loopless paths. *Networks* **16** 149-160.

[23] Pollack, M. (1961). The *k*th best route through a network. *Operations Research* **9** 578-580.

[24] Sedeño-Noda, A., C. González-Martín. 2006. Shortest Path Simplex Algorithm with a Multiple Pivot Rule. *Tecnival Report nº 2, Departamento de Estadística, Investigación Operativa y Computación*.

[25] Yen, J. Y. 1971. Finding the *K* shortest loopless paths in a network. *Management Science* **17** 712-716.

[26] Yen, J. Y., 1972. Another algorithm for finding the *K* shortest loopless network paths. *In Proc. of 41st Mtg. Operations Research Society of America* **20**.