An Algorithm for Calculating k Shortest Paths with a Maximum Number of Arcs

by

Teresa Gomes

Lúcia Martins

José Craveirinha

INESC – Coimbra

Research Report ET-N6
May 2000

Work partially supported by FCT, project PRAXIS/P/EEI/13219/1998, *Um estudo sobre encaminhamento dinâmico multi-objectivo e dependente do estado em redes multi-serviço.*

An Algorithm for Calculating k Shortest Paths with a Maximum Number of Arcs

Teresa Gomes, Lúcia Martins and José M. F. Craveirinha

Departamento de Engenharia Electrotécnica,
Pólo II da Universidade de Coimbra,
Pinhal de Marrocos, 3030 COIMBRA, Portugal.

INESC-Coimbra, Rua Antero de Quental 199,
3000-033 COIMBRA. Portugal.

e-mail: teresa@dee.uc.pt, lucia@dee.uc.pt, jcrav@dee.uc.pt

17th May 2000

Abstract

In multiexchange telecommunication networks the problem of calculating shortest paths involves, in general, constraints on the maximum number of arcs in any path. In highly connected networks the number of available paths between pairs of nodes is very high and if no additional constraints are introduced, shortest paths (for a given objective function) may occur having a significant number of arcs.

In this paper we present an efficient algorithm for calculating the k shortest loopless paths wiht a a maximum number of arcs per path, derived from the MPS algorithm [5, 4] for calculating the k shortest loopless paths.

The proposed algorithm will be compared with the original MPS algorithm when both procedures are used for enumerating the k shortest loopless paths with a maximum number of arcs, in order to show the significant efficiency improvement obtained from the algorithm. The comparison will be made in terms of the total number of generated paths in each algorithm.

Keywords: Routing, shortest paths enumeration.

1 Introduction

The calculation of the K-shortest paths in a network has many applications in science and engineering, namely as a solution to a sub-problem (see ex. [3, 2]). An interesting application in telecommunications as sub-procedure in a multi-objective routing model was recently proposed [1]. In a telecommunication multiexchange network routing context, the determination of of optimal/sub-optimal loopless paths (according to pre-defined criteria) is normally constrained by the maximum number of arcs which may constitute any path. It must be noted that in highly connected networks the number of available paths between pairs of nodes is very high and if no additional constraints are introduced, shortest paths (for a given objective function) may occur having a significant number of arcs.

A first (simplistic) approach to the K-shortest loopless path problem with a maximum (D) number of arcs would be to generate the sequence of the shortest paths, using a very efficient algorithm , such as the one proposed by Martins *et al.* [4] and then to select a subset of K paths by ignoring those which have more than D arcs. As an alternative an algorithm is proposed in this paper for solving this K shortest path constrained problem, which is based on the algorithm for loopless paths in [4] which is a variant of the algorithm in [5]. It will be shown that the proposed algorithm is much more efficient than the former (simplistic) approach since it prevents, by construction, the enumeration of paths with more tan D arcs, therefore reducing the number of generated paths which would be disregarded later on.

This paper is organised as follows. In section 2, beyond reviewing the definitions and notation used in [5] some new definitions will be introduced. The foundations of the algorithm will be presented in section 3. The application of the algorithm to various network examples and its comparison with the simple method of rejecting the paths with more than D arcs in the MPS algorithm, will be performed in section 4. Finally some conclusion on this work and the envisaged application of the algorithm in a bi-objective routing model for telecommunication networks, will be briefly mentioned.

2 Definitions and notation

This section begins by reviewing the basic definitions and notation used in [5]. Let $(\mathcal{N}, \mathcal{A})$ denote the graph defining a network[1] topology, where $\mathcal{N} = \{v_1, v_2, \ldots, v_n\}$ (or $\mathcal{N} = \{1, 2, \ldots, n\}$ to simplify) is the finite set of nodes (or vertices) and $\mathcal{A} =$

[1]The term network will be reserved to designate a broader entity which includes the graph $(\mathcal{N}, \mathcal{A})$ as a fundamental descriptive structure – *e.g.* a telecommunication multiexchange network.

$\{a_1, a_2, \ldots, a_m\}$ is the finite set of arcs or edges, such that each arc is a pair of nodes. If all the pairs are ordered (unordered) the graph is said to be *ordered* (*unordered*). Hereafter and unless something explicit is said, the graph is supposed to be ordered. A path p from s to t ($s, t \in \mathcal{N}$) is defined by an alternating sequence of nodes and arcs, $\langle s = v'_1, a'_1, v'_2, \ldots, a'_{r-1}, v'_r = t \rangle$, where: $a'_k \in \mathcal{A}$ for any $k = 1, \ldots, r-1$ and $v'_k \in \mathcal{N}$ for any $k = 1, \ldots, r$; $a'_k = (v'_k, v'_{k+1})$ for any $k = 1, \ldots, r-1$. The *value* (or *cost*) $c(p)$ of a path p is obtained by associating a cost $c_{ij} \in \mathbb{R}$ with each arc (i, j):

$$c(p) = \sum_{(i,j) \in p} c_{ij} \tag{1}$$

A path from s to t in $(\mathcal{N}, \mathcal{A})$ is a *loopless path* iff all its nodes are different. A *cycle* is a path from a node to itself such that all nodes are different except the first which is identical to the last one. A cycle C is negative iff $c(C) = \sum_{(i,j) \in C} c_{ij} < 0$ and *absorbent* iff $c(C) \leq 0$. Let p_{ij} be a path from i to j; the *concatenation* of paths p_{ij} and p_{jl} is the path, $p_{ij} \diamond p_{jl}$, from i to l, which coincides with p_{ij} from i to j and with p_{jl} from j to l.

Let $P^K = \{p_1, \ldots, p_K\}$ be the set of the K shortest (unconstrained) loopless paths between a given pair of nodes. An algorithm enabling to construct $\mathcal{T}$, a *pseudo-tree of the K-shortest paths* (from s to t), algorithm T1, is given in [5] and will be used in this paper (see the Appendix).

The node v_k, for any $k \in \{1, \ldots, K\}$ given from algorithm T1 is designated as *deviation node* of p_k; the arc of p_k the tail of which is v_k is designated as *deviation arc* and the sub-path from v_k to t in p_k is the *deviation path* of p_k, $p^k_{v_k t}$. Note that although nodes are repeated in $\mathcal{T}$, they are all considered as being different in the pseudo-tree. Let $\mathcal{T}_t$ designate a tree where there is a unique path from any node i to t (tree rooted at t) and $\pi_i(\mathcal{T}_t)$ denote the cost of the path p_{it} in $\mathcal{T}_t$; the *reduced cost* $\bar{c}_{i,j}$ of arc $(i, j) \in \mathcal{A}$ associated with $\mathcal{T}_t$ is $\bar{c}_{ij} = \pi_j(\mathcal{T}_t) - \pi_i(\mathcal{T}_t) + c_{ij}$.

Next additional notation and definitions used in the proposed algorithm will be introduced. Let $|p|$ denote the number of arcs in path p, from s to t, or *path length*. The notation ${}^D p$ will designate a path such that ${}^D p = p$ if $|p| \leq D$; otherwise ${}^D p$ is the sub-path with origin s and D arcs. If $D = 0$ then ${}^D p$ is the empty set $\emptyset$. Let p be a path which contains nodes v_i and v_j. The *distance from v_i to v_j in p* will be given by the number of arcs in p, from v_i to v_j; if v_i, v_j, are extreme nodes of an arc then the distance is 1; the distance of a node to itself is zero. Let p be a path from s to t, which contains v_i; the *depth of v_i in p* is is given by the distance from s to v_i in p and will be designated by $d_p(v_i)$. Let $\mathcal{T}^*_t$ be the tree of the shortest paths from all nodes to t. The *height of node v_i in $\mathcal{T}^*_t$* is given by the distance from v_i to t and is designated

by $h_{\mathcal{T}_t^*}(v_i)$; $p^*_{v_j t} \leftarrow$ represents the shortest path from v_j to t in $\mathcal{T}_t^*$. Let the set of arcs $\mathcal{A}$ of $(\mathcal{N}, \mathcal{A})$ be written in terms of $A(k)$, the set of arcs the tail node of which is $v \in \mathcal{N} = \{1, 2, \ldots, n\}$, *i.e.* $\mathcal{A} = A(1) \cup A(2) \cup \ldots \cup A(n)$ such that $A(i) \cap A(j) = 0$ for any $i \neq j$ $(i, j \in \mathcal{N})$. Let $a'_k \in A(\xi)$ and $a'_l \in A(\theta)$. Assuming that $\xi \neq \theta$ an order relation '<' is defined for the arcs such that $a'_k < a'_l$ iff $\xi < \theta$. Moreover if $\xi = \theta$ then $a'_k < a'_l$ if $\bar{c}(a'_k) \leq \bar{c}(a'_l)$. This means that the set A is sorted in such a way that for any two arcs $(k, j), (i, j) \in \mathcal{A}$, $(k, j) < (i, l)$ if $k < i$ or $(k = i$ and $\bar{c}_{kj} \leq \bar{c}_{il})$. The resulting set $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ is said to be in the *sorted forward star form.*

3 The algorithm

Let one consider that only shortest paths with length less than or equal to D, be acceptable. Then the *K-shortest path constrained problem* which we are seeking to solve for two given nodes s, t is equivalent to determine a set $P_D^K = \{p_1, p_2, \ldots, p_K\} \subset P \equiv P_{st}$ where P_{st} is the set of the paths which can be defined from s to t in $(\mathcal{N}, \mathcal{A})$ such that:

1. $|p_k| \leq D, \quad \forall k \in \{1, 2, \ldots, K\}$
2. $c(p_k) \leq c(p_{k+1}), \quad \forall k \in \{1, 2, \ldots, K\}$
3. $c(p_k) \leq c(p)$ for any p: $|p| \leq D \wedge p \in P - P_D^K$

The proposed algorithm has the additional (desirable) property that p_k is identified just before p_{k+1} for any $k \in \{1, 2, \ldots, K-1\}$.

Let $\mathcal{T}_k$ designate the pseudo-tree of the k-shortest loopless paths from s to t, with a maximum of D arcs and X the set of the candidate paths to the following shortest path in the context of the algorithm MPS in [5]. The justifications underlying the proposed variant to the algorithm MPS in the version in [4] are now presented.

- Let p_u be the shortest path in X (that is the u-th to be selected) and assume that the length of p_u is greater than D – therefore this path is of no interest as a final result of the algorithm. Let v_u be the deviation node of p_u which is at a distance d_u from s.

  Then, there are two scenarios:

  - $d_u < D$, in which case the paths eventually generated from this deviation node v_u or from the nodes v in p_u following v_u are at a depth $d_{p_u}(v) \leq D-1$ and therefore should be generated and stored in X since the deviation nodes of such paths are at a depth less than D.

- $d_u \geq D$ implies that p_u should never have been placed in the candidates path set since it of no interest as a final result and also all the paths obtained from p_u would have length greater than D. In fact if at a given step of the algorithm the deviation node of a path q to be constructed (such that $q_{sv} = p^u_{sv}$, where p^u_{sv} is the sub-path of p_u from s to v) is at a depth D or greater than D, this new path does not need to be constructed for inclusion in X. This is because on the one hand it would not contribute to the identification of paths of interest and on the other hand its exclusion from X does not raise any limitation concerning the generation of the other paths that need to use some of its nodes as deviation nodes. As a matter of fact all nodes of q relevant for this purpose (at most the first $D-1$ nodes) are common to the path p_u and should already have been stored in $\mathcal{T}_u$. Therefore the paths p with deviation node v such that $d_p(v) \geq D$ will never be generated and included in X.

- Let p_u be a path of length D.

  By definition the deviation node of p_u could never be node t. If the deviation node of p_u is at depth $D-1$, then from the MPS algorithm, v_u could give origin to new paths of length always greater than D, hence it is not worth to generate them. In fact such paths would have a deviation node at a depth greater than or equal to $D-1$; hence would always have length greater than D, as well as all the paths obtained from them.

  If the deviation node v_u has a depth $d_u < D-1$ then the path eventually obtained from v_u and the paths eventually obtained from the nodes following v_u with depth less than $D-1$, should be generated since their deviation nodes are at a depth less than $D-1$.

- Let p_u be a path from s to t of length d_u less than D. Then all the paths that may be obtained from p_u should be generated and stored in X. This results from the fact that the deviation nodes of such paths could always be at a depth less than D and potentially could give origin to paths with length less than or equal to D.

The inverse Dijkstra algorithm used as a starting point of the algorithm will be adapted to associate each node with its height in the shortest paths tree from all nodes to t. This is easily obtained by giving to t the height 0 and establishing that whenever

the successor of a node v_i is changed, being for example v_j, then the height associated with v_i is also changed and becomes the height of v_j plus 1.

A final note on the used notation. Given a path p (from s to t) and a node v in p, p_{sv} represents the sub-path of p from s to v and p_{vt} represents the sub-path of p from v to t.

Next the algorithm for generating the K-shortest paths of length less or equal to D, is formalised.

Algorithm KD

1. Input: the representation of the graph $(\mathcal{N}, \mathcal{A})$ and arc costs c_{ij}.
2. By using Dijkstra inverse algorithm (including the node heights calculations), obtain the shortest paths from every node to t, hence constructing the *shortest tree rooted at* t, $\mathcal{T}_t^*$.
3. Calculate the reduced cost $\bar{c}_{ij}$ for every $(i,j) \in \mathcal{A}$.
4. Rearrange the arcs of $(\mathcal{N}, \mathcal{A})$ in the sorted forward star form (for the computed $\bar{c}_{ij}$, leading to $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ such that for any $h \in \{1, 2, \ldots, m-1\}$, $\bar{c}_{a_h} \leq \bar{c}_{a_{h+1}}$ if $v = \text{tail}(a_h) = \text{tail}(a_{a_{h+1}})$)[2]
5. $p \leftarrow$ shortest path from s to t ($p \in \mathcal{T}_t^*$).
6. $k \leftarrow 0$
7. $X \leftarrow \{p\}$ (X is the set of paths that are candidates to shortest path).
8. `While` $(k < K \wedge X \neq \emptyset)$ `do`
   (a) $p \leftarrow$ shortest path in X
   (b) $X \leftarrow X - \{p\}$
   (c) Calculate $|p|$
   (d) `If` $|p| \leq D$ and p has no loops `then`
      i. $k \leftarrow k + 1$
      ii. $p_k \leftarrow p$ (the k-th path which has at most D arcs, was found)
   `EndIf`
   (e) Let v be the deviation node of p
   (f) $p_{sv} \leftarrow$ sub-path of p from s to v

[2] In case $\bar{c}_{a_h} = \bar{c}_{a_{h+1}}$, and a_h is the the first element of $A(v \neq t)$, then a_h is also the arc which belongs to $\mathcal{T}_t^*$. In other words, the first element of each non empty $A(v)$, $v \neq t \wedge v \in \mathcal{N}$ must belong to $\mathcal{T}_t^*$.

(g) `If` ($|p| = D$ and $|p_{sv}| < D-1$)
`then` $l \leftarrow D-2$ (Try to generate new paths[3])
`else If` ($|p| \neq D$ and $|p_{sv}| < D$)
`then` $l \leftarrow \min(|p|-1, D-1)$ (Try to generate new paths[4])
`else` $l \leftarrow -1$[5] (No path will be generated from p)
EndIf EndIf

(h) `If` $l \neq -1$ `then`

i. $v_i \leftarrow v$

ii. Concluded $\leftarrow$ False (Possibly a path will be placed in X)

iii. `Repeat`

A. $a_h \leftarrow$ the arc of p the tail of which is v_i

B. $p_{sv_i} \leftarrow$ sub-path of p from s to v_i

C. `While` (v_i is the tail of a_{h+1}) and (a_{h+1} forms a loop with p_{sv_i}) `do`
$h \leftarrow h+1$
`EndWhile`

D. `If` v_i is the tail of a_{h+1} `then`
- $v_j \leftarrow$ head of a_{h+1}
- $p^*_{v_j t} \leftarrow$ shortest path from v_j to t in $\mathcal{T}^*_t$
- $X \leftarrow X \cup \{p_{sv_i} \diamond \langle v_i, a_{h+1}, v_j \rangle \diamond p^*_{v_j t}\}$

`EndIf`

E. `If` $d_p(v_i) = l$[6]
`then` Concluded $\leftarrow$ True
`else` $v \leftarrow$ following node in ${}^l p$ (truncated path p after l arcs)
`EndIf`

`Until` (p_{sv_i} has a loop) or (Concluded)[7]

`EndIf`($l \neq -1$)

`EndWhile`

4 Analysis of Results

Having in mind to compare the algorithm KD proposed in section 3 , (which will be designated hereafter by "KD") with results from the algorithm MPS [4] (hereafter designated by "MPS") for the purpose of generating the K-shortest loopless paths

[3]Such paths will diverge from p from the deviation node of p onwards and/or from the nodes of p placed after v up to a distance $D-2$ from s.

[4]Such paths will diverge from p from the deviation node of p onwards and/or from the nodes of p placed after v up to a distance $D-1$ (if $|p| < D$) or $|p|-1$ (if $|p| > D$) from s.

[5]Since from path p it is not possible to obtain any path with less than $D+1$ arcs.

[6]This means that v_i is the last node of ${}^l p$.

[7]Since ${}^l p$ does not contain t anymore the loop only terminates when all the nodes of ${}^l p$ have been used, which is signalled by making the boolean variable Concluded equal to True, or when the sub-path p_{sv_i} presents a loop.

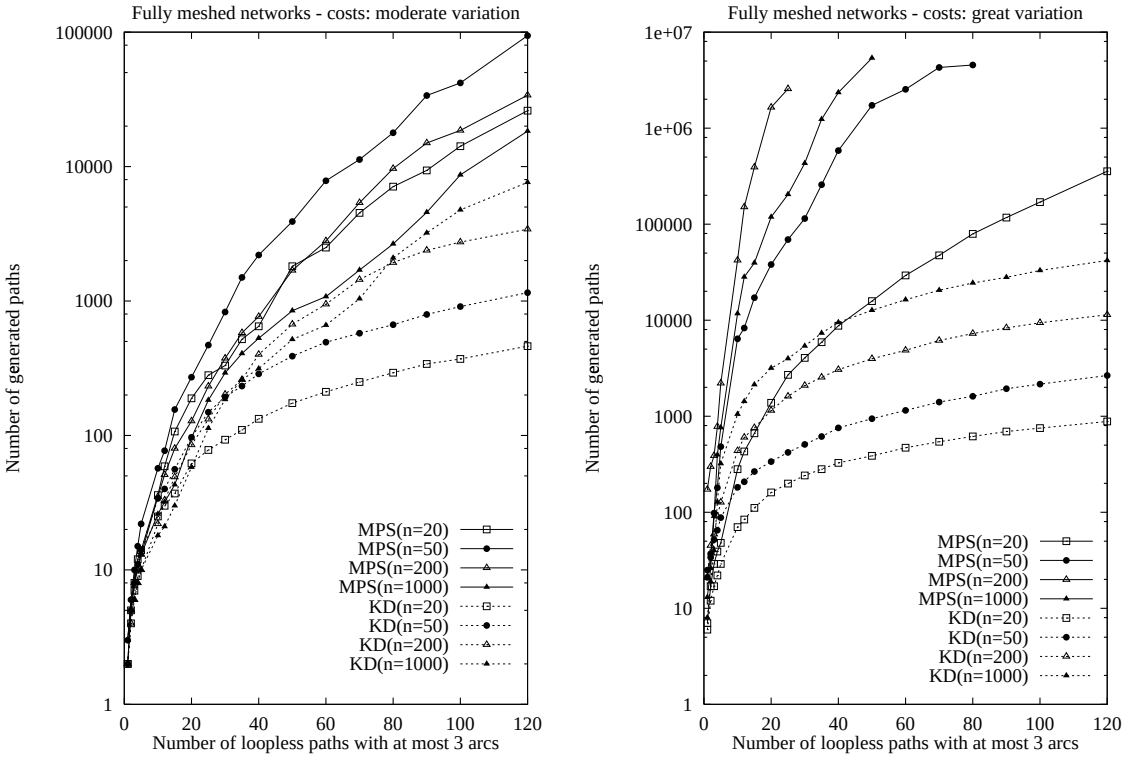

Figure 1: Comparison of the algorithms for $D = 3$; **(a)** $n = 20, 50$, $c_{ij} \in [1, 10]$; $n = 200$, $c_{ij} \in [1, 20]$ and $n = 1000$, $c_{ij} \in [1, 40]$ **(b)** $n = 20$, $c_{ij} \in [1, 50]$ and $n = 50, 200, 1000$, $c_{ij} \in [1, 150]$

with a maximum of D arcs, numerous computational experiments were carried out. Strongly meshed networks and fully meshed networks (corresponding to strongly or fully connected graphs) where considered[8], with 20, 50, 200 and 1000 nodes. The comparison measure was the total number of paths generated by each algorithm until the K-shortest path with at most D arcs was obtained. To facilitate the comparison, values were obtained by averaging the number of generated paths corresponding to solve the problem for pairs of origin-destination nodes in a given network and at least in two networks obtained from different random number sequences, which determine the values of the arc cost (in a given interval) and also the network arcs in the case of strongly meshed networks.

A major factor which influences the relative performance of MPS is the range of variation of arcs costs. For this reason results were presented for two ranges of variation: a moderate variation, from 1 to 10 in 20 node networks up to 1 to 40 in 1000 node

[8]The used program for network generation was kindly borrowed by José Luís Santos.

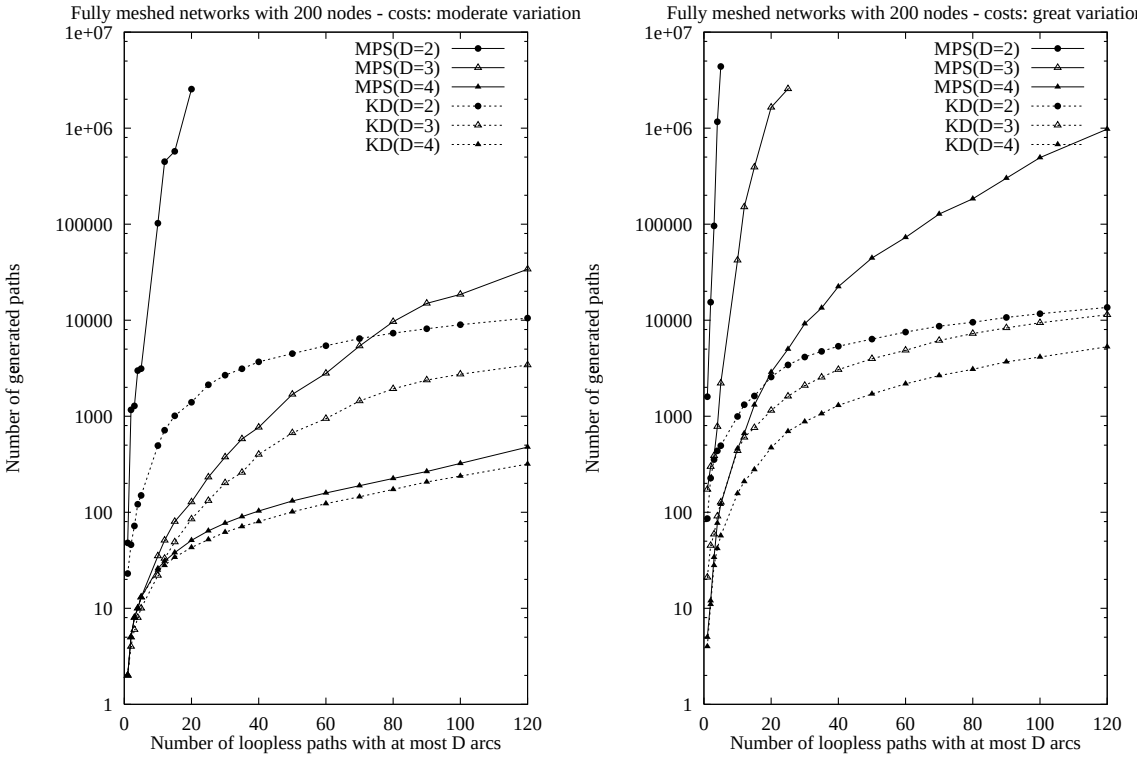

Figure 2: Comparison of the algorithms for fully meshed networks with 200 nodes and and $D = 2, 3, 4$; (a) arc costs $\in [1, 20]$ (b) arc costs $\in [1, 150]$

networks and a second range of larger relative variations of 1 to 50 in 20 node networks and 1 to 150 for the remaining networks. In MPS the higher is the cost variation, the higher is the average number of arcs of the shortest paths, hence the poorer is the relative performance of MPS, as illustrated in figures 1, 2 and 3.

Some of the most significant results are condensed in figures 1, 2 and 3. The presented curves represent averages over the total number of paths generated for obtaining up to 120 shortest paths (whenever they exist) between two pairs of nodes with a maximum of $D = 2, 3$ or 4 arcs. Note that these are the values of D (specially $D = 2, 3$) which are of interest to applications in route generation algorithms in multiexchange networks. The considered number of generated paths is the number of paths inserted in the set X of candidate paths in the course of algorithm execution. To prevent excessive running times in some cases (due to swapping) any of the algorithms is stopped when the 10^7-th path is generated even if the last shortest path with up to D arcs hasn't been obtained. This limiting situation occurred some times with the MPS algorithm, but never with the KD algorithm.

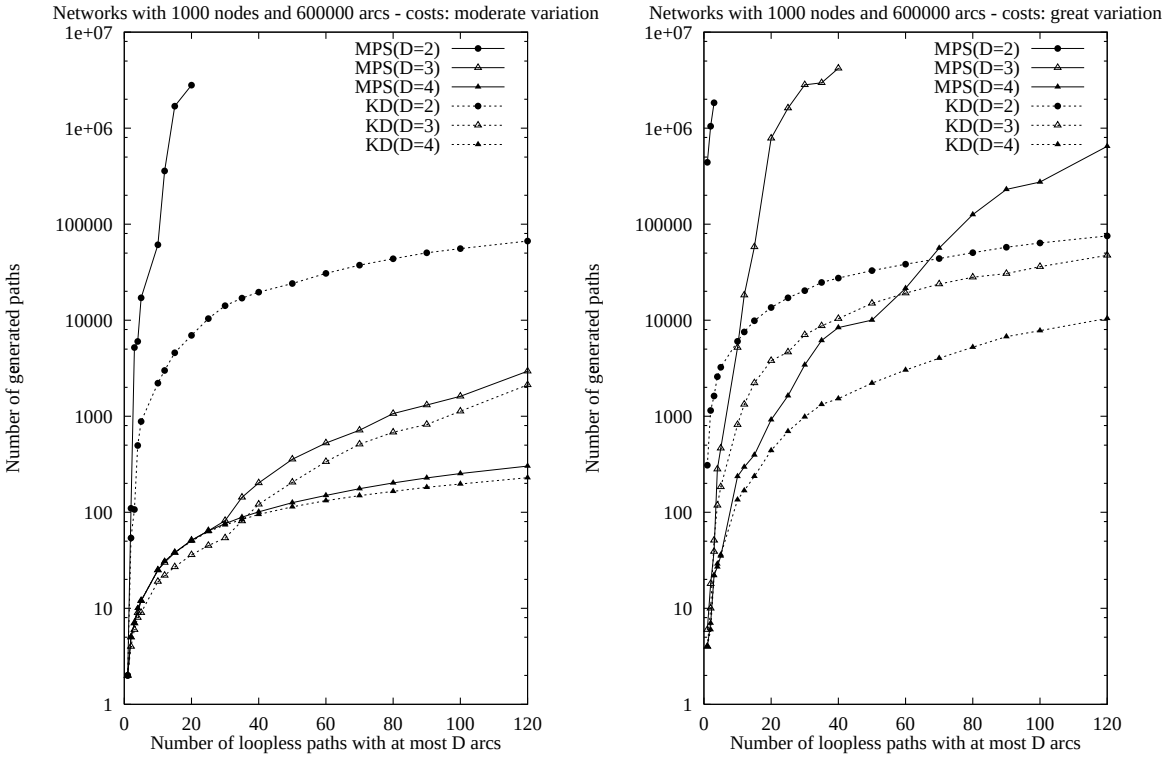

Figure 3: Comparison of the algorithms for non-fully meshed networks with 1000 nodes and 600000 arcs, $D = 2, 3, 4$; (a) arc costs $\in [1, 20]$ (b) arc costs $\in [1, 150]$

It is apparent from the results the quite significant advantage of KD with respect to MPS in most cases. This highly improved efficiency of KD is expectable having in mind that KD does not generate unnecessary paths (that is with more than D arcs – when such paths are of no potential interest) while it makes the most of the inherent efficiency of MPS. Other advantage of KD stems from the fact that it manages to identify the phase in which there are no more paths with the desired number of arcs. This is particularly important in non fully meshed networks where one seeks to obtain paths with 2 our 3 arcs, a common situation in telecommunication networks.

In general the total number of paths generated by MPS is closer to the equivalent number for KD, when the maximum number of arcs is higher and the relative variation of the arc costs is smaller. This stems from the fact that, in these cases, almost all the first shortest paths generated by MPS have a number of arcs less than or equal to D.

It was also verified that the relative performance of the algorithms did not depend so much on the network dimensions than on the cost variations. This may be appreciated from figures 1(a) and 1(b) where results are presented for several fully meshed networks

and for two ranges of cost variations, with $D = 3$.

5 Applications and conclusions

In some applications of the K shortest path problem an important constraint is the maximum number of arcs in any chosen path. In particular, for multiexchange telecommunication networks, the determination of optimal or suboptimal (according to some criteria) paths which may be used for routing calls between pairs of nodes (associated with exchanges in terms of network representation) imposes, for technical reasons, that only paths with up to a certain number of arcs, may be considered. This has to do with the necessity of simplifying control and signalling functions and prevent undesirable situations from the point of view of the traffic flow distributions. Note also that multiexchange telecommunication networks are usually either fully meshed or strongly meshed and technology enables the utilisation of sophisticated routing mechanisms. Such mechanisms usually require frequent recalculation (for different network conditions *e.g.* related to traffic loads or channel occupations) of shortest paths for each pair of origin-destination nodes. An example of this type of application environment for this constrained K-shortest path problem is a multi-objective routing model proposed in [1]. The basis of this model is to consider two objective functions (to be minimised) in the routing model such as cost and blocking probability or cost and delay. This model involves the selection of non-dominated solutions (paths) of a bi-objective problem taking into account some preference criteria defined on the objective function space. The algorithm used for tacking this problem uses repeatedly the K-shortest path algorithm [4] and the authors are developing a new variant of the model enabling the choice of alternative paths when for example the first selected path is blocked (ie. at least one arc is unavailable), a technique that is known as alternative routing, also considering as explicit constraint the maximum number D of arcs per path ($D = 2, 3, 4$).

An algorithm was therefore proposed in this paper for solving the K-shortest path problem with a constraint on the number of arcs per path, which is based on the MPS algorithm for loopless paths in [4] (which is a variant of the algorithm in [5]). It was shown that the proposed algorithm (algorithm KD) is much more efficient in most cases than the simplistic approach of ignoring the paths selected by MPS with more than D arcs. This was confirmed by extensive computational results some of which were discussed in the paper. This highly improved efficiency of KD is expectable having in mind that KD does not generate unnecessary (that is with more than D arcs – when such paths are of no potential interest) while it makes the most of the inherent

efficiency of MPS. Other advantage of KD stems from the fact that it manages to identify the phase in which there are no more paths with the desired number of arcs. This is particularly important in non fully meshed networks where one seeks to obtain paths with 2 our 3 arcs, a common situation in telecommunication networks.

The experimental results also suggest that in general the total number of of paths generated by the two algorithms is closer when the maximum number of arcs is higher and the relative variation of the arc costs is smaller. This stems from the fact that, in these cases, almost all the first shortest paths generated by MPS have a number of arcs less than or equal to D.

Acknowledgements: The authors are very grateful to Professor Ernesto Martins, Marta Pascoal and José Luís Santos for their explanations concerning the MPS algorithm and its implementation.

A Algorithm T1

Algorithm T1: Constructing the pseudo-tree of the K-shortest paths[5]

$\mathcal{T} \leftarrow 0$

$k \leftarrow 1$

`While` $k \leq K$

`do begin`

    $arc \leftarrow$ first arc p_k

    $head \leftarrow$ head node of arc

    $p \leftarrow$ sub-path of p_k from s to $head$

    `While` $p \in \mathcal{T}$

    `do begin`

        $arc \leftarrow$ following arc in p_k

        $head \leftarrow$ head node of arc

        $p \leftarrow$ sub-path of p_k from s to $head$

    `end`

    $v_k \leftarrow$ tail node of arc

    $p^k_{v_k t} \leftarrow$ sub-path of p_k from v_k to t

    $p^k_{s v_k} \leftarrow$ sub-path of p_k from s to v_k

    $\mathcal{T} \leftarrow \mathcal{T} \cup p^k_{v_k t}$ /* In such a way that $p^k_{s v_k} \diamond p^k_{v_k t}$ is a path of $\mathcal{T}$ */

    $k \leftarrow k + 1$

`end`

References

[1] C. Hengeller Antunes, J. Craveirinha, J. Clímaco, and C Barrico. A multiple objective routing algorithm for integrated communication networks. In *Teletraffic Engineering in a Competitive World – ITC 16*, pages 1291–1300. Elsevier, 1999.

[2] J. C. N. Clímaco and E. Q. V. Martins. A bicriterion shortest path algorithm. *European Journal of Operational Research*, (11):399–404, 1982.

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

[4] E. Martins, M. Pascoal, and J. Santos. An algorithm for ranking loopless paths. Technical Report 99/007, CISUC, 1999. http://www.mat.uc.pt/~marta/Publicacoes/mps2.ps.gz.

[5] E. Martins, M. Pascoal, and J. Santos. Deviation algorithms for ranking shortest paths. *International Journal of Foundations of Computer Science*, 10(3):247–263, 1999.