Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computers & Operations Research 32 (2005) 509–520

computers & operations research

www.elsevier.com/locate/dsw

An algorithm for ranking quickest simple paths

Marta M.B. Pascoal[a,d,*], M. Eugénia V. Captivo[b], João C.N. Clímaco[c,e]

[a]*Centro de Informática e Sistemas, Polo I da Universidade de Coimbra, Apartado 3008, Coimbra 3001-454, Portugal*
[b]*DEIO-CIO, Faculdade de Ciências, Universidade de Lisboa, Bloco C2, Campo Grande, Lisboa 1749-016, Portugal*
[c]*Instituto de Engenharia de Sistemas e Computadores—Coimbra, Rua Antero de Quental, 199, Coimbra 3000-033, Portugal*
[d]*Departamento de Matemática, Polo I da Universidade de Coimbra, Apartado 3008, Coimbra 3001-454, Portugal*
[e]*Faculdade de Economia da Universidade de Coimbra, Avenida Dias da Silva, 165, Coimbra 3004-512, Portugal*

Available online 10 September 2003

Abstract

In this paper, an algorithm for ranking loopless paths in undirected networks, according to the transmission time, is presented. It is shown that the worst-case computational time complexity of the algorithm presented is $\mathcal{O}(Kr(m+n\log n))$, which is also the best-known complexity to solve this problem. The worst-case memory complexity is $\mathcal{O}(Kn)$, which improves the existing algorithms. Finally, comparative computational results, with other algorithms for the same problem, are reported.
© 2003 Elsevier Ltd. All rights reserved.

Keywords: Graph theory; Network; Quickest path; Simple path; Paths ranking

1. Introduction

The optimal path problem is very well known and many variants of this problem have been proposed in the literature, the shortest path problem being one of the most studied. The quickest path problem is an optimal path problem where it is intended to compute a path that sends a given amount of data from an initial node to a terminal node with minimum transmission time.

Let $(\mathcal{N},\mathcal{A})$ be a network with n nodes and m arcs where, for any given $(i,j)\in\mathcal{A}$, $l_{ij}\in\mathbb{R}_0^+$ denotes the lead time for traversing (i,j), and $c_{ij}\in\mathbb{R}^+$ its capacity per time unit. Given $\sigma\in\mathbb{R}^+$, the total

[*] Corresponding author. Departamento de Matemática, Polo I da Universidade de Coimbra, Apartado 3008, Coimbra 3001-454, Portugal. Tel.: +351-239-791150; fax: +351-239-832568.
E-mail addresses: marta@mat.uc.pt (M.M.B. Pascoal), mecaptivo@fc.ul.pt (M.E.V. Captivo), jclimaco@inescc.pt (J.C.N. Clímaco).

time to transmit σ units of data throughout a path p is defined by $T(p) = l(p) + \sigma/c(p)$, where $l(p) = \sum_{(i,j)\in p} l_{ij}$ and $c(p) = \min_{(i,j)\in p} \{c_{ij}\}$. Let $s, t \in \mathcal{N}$ (with $s \neq t$) be the initial and terminal nodes of $(\mathcal{N}, \mathcal{A})$, respectively, and denote the set of paths from s to t by $\mathcal{P}$. Then the quickest path problem consists in $\min_{p\in\mathcal{P}} \{T(p)\}$. This problem was introduced by Moore [1] in 1976, and the first algorithm to solve it was presented in 1990 by Chen and Chin [2]. These authors noticed that with a fixed capacity value the quickest path problem is reduced to a shortest path problem, which lead them to propose an extension of the original network, with as many levels as the number r of distinct capacity values. Each path between specific nodes in the modified network corresponds to a path with a certain capacity in the original one; therefore, capacity values can be ignored and the solution of the initial problem can be found with a labelling algorithm, since it is a shortest path in the augmented network. This network contains rn nodes and rm arcs; therefore, the space complexity of the algorithm by Chen and Chin is of $\mathcal{O}(rm + rn)$, while its time complexity is of $\mathcal{O}(rm + rn \log n)$. Later, Rosen et al. [3] noticed that the extended network can be replaced by a sequence of, at most, r networks with increasing capacity lower bound (therefore, subnetworks of the original one), the shortest path problem resolution on the modified network being replaced by several shortest path problems resolution, one in each of those subnetworks. The quickest path is the one with minimum total transmission time. A different approach was introduced by Martins and Santos [4] in 1997, whom studied the problem from a biobjective point of view. The result is an algorithm very similar to the one by Rosen et al., but where the shortest paths with maximum capacity are computed, instead of the shortest paths, trying to solve less problems. When the worst case is considered, these two algorithms improve the space complexity of Chen and Chin's algorithm to $\mathcal{O}(m + n)$, although they still have the same time complexity. Other variants of the quickest path problem have also been studied. We refer to the all-pairs quickest path problem [5], and the K quickest path problem [6], where paths may contain repeated nodes.

As for the shortest path problem, the quickest path problem can be generalized to enumerate the K quickest paths from s to t, or the K quickest loopless paths (or simple paths) from s to t, given an integer $K > 1$. The present paper deals only with the K quickest loopless paths problem. For short, in the following the term path will be used in place of simple path. Two algorithms to solve this problem are referred to:

- one due to Rosen et al. (very similar to another one, proposed for ranking optimal paths, by Martins and Pascoal [7]), valid for any network and with time complexity $\mathcal{O}(Knr(m + n \log n))$, and space complexity $\mathcal{O}(Kn)$;
- and another one by Chen [8], valid for any type of network, with the same time complexity but $\mathcal{O}(Knr)$ space memory requirements, if using Yen's algorithm [9] to rank shortest paths; and valid only on undirected networks, with time of $\mathcal{O}(Kr(m+n \log n))$ and space of $\mathcal{O}(Knr)$, if using Katoh et al.'s algorithm [10] to rank shortest paths.

In this paper, the algorithms of Rosen et al. and of Katoh et al. are merged, in order to obtain a new one, able to rank quickest paths in undirected networks with $\mathcal{O}(Kr(m+n \log n))$ time complexity and $\mathcal{O}(Kn)$ space complexity. In Section 2, the algorithms of Rosen et al. and Katoh et al. are reviewed, the new algorithm is introduced, and its theoretical complexity is studied. Section 3 is dedicated to computational results and conclusions are presented in Section 4.

2. The K quickest loopless paths problem

Let $\mathrm{sub}_p(x, y)$ denote the subpath of p from node x to node y, and let $p \diamond q$ denote the concatenation of paths p and q, that is, the path formed by p and followed by q.

In [9], Yen presented an algorithm to rank the shortest paths. He suggested the use of a set X of paths, each one candidate to the kth shortest path, for some $k \in \{1, \dots, K\}$. Thus, set X is initialized with the best path (regarding the cost function) and, after that, the best element in X is selected and analysed, in order to find new paths to insert in X. The paths selected correspond to $p_1, p_2, \dots,$ and the algorithm halts when K paths are found. Given one of those paths $p_k = \langle s = v_1, v_2, \dots, t = v_{\ell(p_k)} \rangle$, the new candidates for following shortest paths are obtained from the remaining set of paths,

$$\mathscr{P}^j(v_{d(p_k)}) - \{p_k\} = \bigcup_{i=d(p_k)}^{\ell(p_k)-1} \mathscr{P}^k(v_i)$$

and solving the shortest path problem in every $\mathscr{P}^k(v_i)$. $v_{d(p_k)}$ is the deviation node of p_k, that is, the node where p_k separates from the path p_j that generated it. $\mathscr{P}^k(v_i)$ is the set of paths, different from $p_1, \dots, p_k$, of the form $q_i = \mathrm{sub}_{p_k}(s, v_i) \diamond q$, where q is a path from v_i to t.

To exemplify, consider the paths from 1 to 5 in the network depicted in Fig. 1, and $p_1 = \langle 1,3,4,5 \rangle$, $p_2 = \langle 1,2,4,5 \rangle$. According to Yen's partition, $\mathscr{P} - \{p_1\} = \mathscr{P}^1(1) \cup \mathscr{P}^1(3) \cup \mathscr{P}^1(4)$, which is represented in the first tree in Fig. 2. The different sets in the partition are distinguished by different line styles, the solid one being p_1. Analogously, $p_2 \in \mathscr{P}^1(1)$ and $\mathscr{P}^1(1) - \{p_2\} = \mathscr{P}^2(1) \cup \mathscr{P}^2(2) \cup \mathscr{P}^2(4)$. This partition is depicted in the second tree of Fig. 2.

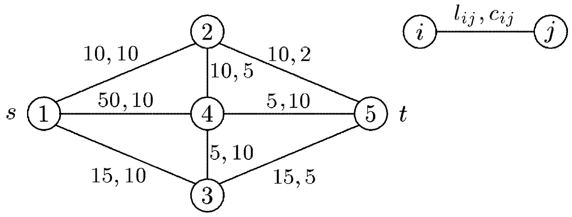
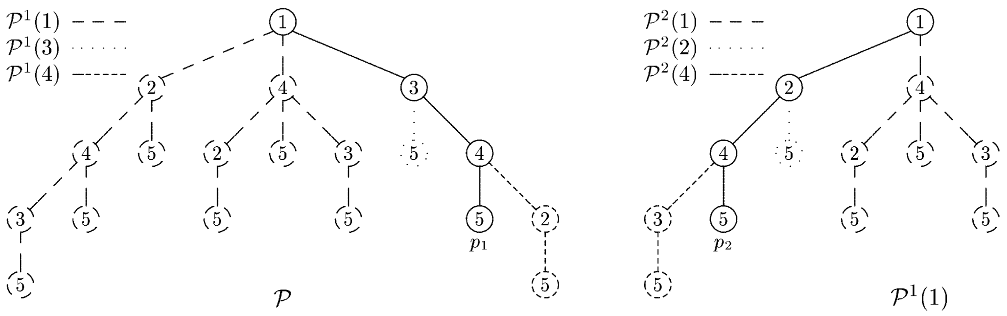


Fig. 1. Network $(\mathcal{N}, \mathcal{A})$.



$\mathcal{P}^1(1)$ - - -
$\mathcal{P}^1(3)$ ......
$\mathcal{P}^1(4)$ ------
p_1
$\mathcal{P}$
$\mathcal{P}^2(1)$ - - -
$\mathcal{P}^2(2)$ ......
$\mathcal{P}^2(4)$ ------
p_2
$\mathcal{P}^1(1)$

Fig. 2. Trees of paths from 1 to 5 in $(\mathcal{N}, \mathcal{A})$.

Table 1
Alternative costs to $(\mathcal{N},\mathcal{A})$

i	1			2		3		4
j	2	3	4	4	5	4	5	5
(l_{ij},c_{ij})	(10,10)	(15,10)	(50,5)	(10,5)	(10,2)	(5,10)	(15,5)	(35,10)

In [3], Rosen et al. use Yen's process, but considering the objective function to be the total transmission time. However, a suitable adaptation has to be made, to compute the quickest path in a given $\mathcal{P}^k(v_i)$. In fact, for Yen's algorithm, i.e. for the shortest path problem, it is enough to

- delete from $(\mathcal{N},\mathcal{A})$ the arcs (v_i,v_{i+1}) and $(v_{d(p_k)},x)\in p_j$, for some $j\in\{1,\dots,k\}$ (so that the new path is not $p_1,\dots,p_k$);
- assure that q_i is loopless, then the nodes in $\text{sub}_{p_k}(s,v_i)$ cannot be repeated; therefore, they are also deleted (except v_i);
- and then determine q, the shortest path from v_i to t (then q_i is the shortest path from s to t).

This last statement is not valid for quickest paths,[1] therefore, analogously to the algorithm they proposed for the quickest path problem, Rosen et al. replaced the resolution of the shortest path problem by the resolution of one shortest path problem in each network, of a sequence obtained by fixing the minimum capacity allowed, based on Theorem 1. Let $(\mathcal{N}_p,\mathcal{A}_p)$ denote the subnetwork of $(\mathcal{N},\mathcal{A})$ without the nodes of p (except the last one), and $(\mathcal{N},\mathcal{A}(w))$ the subnetwork of $(\mathcal{N},\mathcal{A})$ without the arcs with capacity smaller than w. Rosen et al. proved:

Theorem 1. *Let q'_j be the minimum lead time path from v_i to t in $(\mathcal{N}_{\text{sub}_{p_k}(s,v_i)},\mathcal{A}_{\text{sub}_{p_k}(s,v_i)}(c(q'_j)))$ and $p'_j=\text{sub}_{p_k}(s,v_i)\diamond q'_j$, with $j=1,\dots,r$. Then the quickest path in $\mathcal{P}^k(v_i)$ is p^*, such that $T(p^*)=\min_{1\leqslant j\leqslant r}\{T(p'_j)\}$.*

To illustrate this result, consider the graph in Fig. 1 but with the arc lead time and capacity per time unit as in Table 1 and $\sigma=100$. According to Theorem 1, to compute the quickest path from 1 to 5 in $(\mathcal{N},\mathcal{A})$ with initial part $\langle 1,4\rangle$, the shortest path problem from 4 to 5 is solved in $(\mathcal{N},\mathcal{A}(2))=(\mathcal{N},\mathcal{A}),(\mathcal{N},\mathcal{A}(5))$ and $(\mathcal{N},\mathcal{A}(10))$, thus obtaining $q_1=\langle 4,2,5\rangle$ (or $\langle 4,3,5\rangle$), $q_2=\langle 4,3,5\rangle$, $q_3=\langle 4,5\rangle$, respectively. Since $T(\langle 1,4\rangle\diamond q_1)=120$, $T(\langle 1,4\rangle\diamond q_2)=90$ and $T(\langle 1,4\rangle\diamond q_3)=105$, the quickest one is $\langle 1,4,3,5\rangle$.

The two major contributions of Rosen et al.'s algorithm are:

- to notice that the set of paths partition used by Yen does not depend on the objective function;
- to propose a procedure to find the quickest path satisfying specific constraints.

This is the motivation to use an analogous approach for adapting Katoh et al.'s algorithm to the same problem. The major drawback of Yen's algorithm is that, in a worst-case, it demands the

[1] For instance, consider $\sigma=20$ and paths $p=\langle 1,3\rangle$, $p'=\langle 1,2,3\rangle$ and $q=\langle 3,4\rangle$, with $l(p)=l(q)=10$, $l(p')=8$, $c(p)=20$ and $c(p')=c(q)=5$. Then, p is quicker than p', and yet $T(p\diamond q)=24>T(p'\diamond q)=22$.

computation of $n-1$ new paths (the maximum number of arcs in a simple path), for each p_k analysed. In [10], Katoh et al. presented another algorithm for the K shortest paths problem which, in turn, uses a set of paths partition that does not depend on the size of p_k, and allows to rank shortest paths, computing at most three new paths for each p_k.

Let p_k be obtained as the best path in $\mathscr{P}_k^j(v_\delta, v_\gamma)$, where

- p_j is the path analysed to obtain p_k,
- v_δ is the deviation node of another path obtained from p_j, which is the farthest from s and precedes $v_{d(p_k)}$,
- v_γ is the deviation node of another path obtained from p_j, closest to s and that follows $v_{d(p_k)}$.

$\mathscr{P}_k^j(v_\delta, v_\gamma)$ is formed by paths $p = \mathrm{sub}_{p_j}(s, v_\delta) \diamond q$, different from $p_1, \ldots, p_k$, where q is a path from v_δ to t that deviates from p_j before node v_δ. Katoh et al. noticed that

$$\mathscr{P}_k^j(v_\delta, v_\gamma) - \{p_k\} = \mathscr{P}_{k+1}^j(v_\delta, v_{d(p_k)}) \cup \mathscr{P}_{k+1}^j(v_{d(p_k)}, v_\gamma) \cup \mathscr{P}_{k+1}^k(v_{d(p_k)+1}, t).$$

The process to determine the shortest path in each of those subsets, say $\mathscr{P}_k^j(v_x, v_y)$, uses a Yen-like procedure. In fact, that path is $p = \mathrm{sub}_{p_j}(s, v_x) \diamond q$, where q deviates from $\mathrm{sub}_{p_j}(v_x, t)$ before v_y; so, as before, the nodes in $\mathrm{sub}_{p_j}(s, v_x)$ and some arcs of other paths generated from p_j, precedent to v_x, are deleted. Moreover, Katoh et al. introduced a procedure that computes such a q, as long as the network is undirected and $l_{ij} \geqslant 0$, for any $(i,j) \in \mathscr{A}$, based on the computation of two shortest path trees and the analysis of the arcs and nodes in the network. From now on, only undirected networks will be considered. Let $\mathscr{T}_s$ be the tree of the shortest paths from s to any $i \in \mathscr{N}$, and $\mathscr{T}_t$ be the tree of the shortest paths from any $i \in \mathscr{N}$ to t. Denote by $\mathscr{T}_s(i)$ the path from s to $i \in \mathscr{N}$ in $\mathscr{T}_s$, and by $\xi_s(i)$ the index of the node where $\mathscr{T}_s(i)$ deviates from $p^* = \mathscr{T}_s(t) = \mathscr{T}_t(s)$ (analogously for $\mathscr{T}_t(i)$ and $\xi_t(i)$). Katoh et al. proved the following result.

Lemma 1. *Let $p^* = \mathscr{T}_s(t) = \mathscr{T}_t(s)$, and $p^* = \langle s = v_1, v_2, \ldots, v_{\ell(p^*)} = t \rangle$, be the shortest path from s to t in $(\mathscr{N}, \mathscr{A})$. If there exists a path from s to t deviating from p^* before $v_\alpha \in p^*$, then the shortest one is either of type* 1 *or* 2:

Type 1: $\mathscr{T}_s(u) \diamond \mathscr{T}_t(u)$, *with* $\xi_s(u) < \alpha$,
Type 2: $\mathscr{T}_s(u) \diamond (u,v) \diamond \mathscr{T}_t(v)$, *with* $(u,v) \notin \mathscr{T}_s \cup \mathscr{T}_t$ *and* $\xi_s(u) < \alpha$.

As mentioned, the adaptation of this algorithm uses the strategy of Rosen et al., considering the partition of Katoh et al. Moreover, based on Theorem 2, solving the constrained problem of finding the quickest path in $\mathscr{P}_k^j(v_x, v_y)$ is replaced by computing the shortest paths in that set defined over $(\mathscr{N}_{\mathrm{sub}_{p_k}(s,v_x)}, \mathscr{A}_{\mathrm{sub}_{p_k}(s,v_x)}(c_i))$, for any $i = 1, \ldots, r$, where $c_1, \ldots, c_r$ are the distinct values of the arcs capacities.

Theorem 2. *Let q_j' be the minimum lead time path from v_x to t, deviating from* $\mathrm{sub}_{p_j}(v_x, t)$ *before $v_y \in p_j$, in $(\mathscr{N}_{\mathrm{sub}_{p_k}(s,v_x)}, \mathscr{A}_{\mathrm{sub}_{p_k}(s,v_x)}(c(q_j')))$ and $p_j' = \mathrm{sub}_{p_k}(s, v_k) \diamond q_j'$, with $j = 1, \ldots, r$. Then the quickest path in $\mathscr{P}_k^j(v_x, v_y)$ is p^* such that $T(p^*) = \min_{1 \leqslant j \leqslant r} \{T(p_j')\}$.*

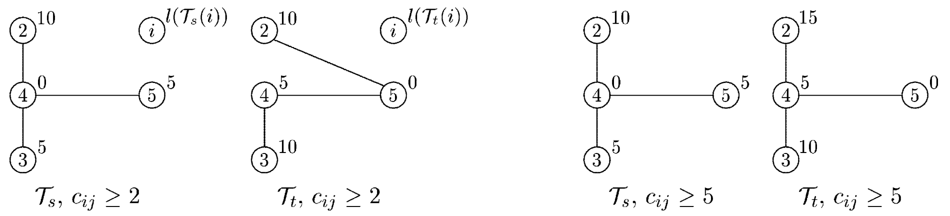


Fig. 3. Trees of the shortest paths from 4 to $i \in \mathcal{N}$, and from $i \in \mathcal{N}$ to 5, in subnetworks of $(\mathcal{N}_{\langle 1,4\rangle}, \mathcal{A}_{\langle 1,4\rangle})$.

As the result, the proof of Theorem 2 is analogous to the one of Theorem 1, which can be found in [3], and therefore is omitted. It remains to answer the question of how to determine the path with minimum lead time, coincident with p_j until a node previous to v_y in each network that has to be considered. It should be noticed that the conditions of Lemma 1 may not be satisfied, since it can happen that $q = \mathrm{sub}_{p_j}(v_x, t)$ is not a path in the network, or that it is not the shortest one. Assuming there exists a path $p = \mathcal{T}_s(t) = \mathcal{T}_t(s)$, the following procedure is suggested:

- If $q \neq p$, then p is the path to be used.
- Else Lemma 1 can be applied.

Recall the network in Fig. 1 with $\sigma = 100$, and consider the path $p = \langle 1,4,5\rangle$. According to Theorem 2, the quickest path from 1 to 5, different from p and with the form $\langle 1,4\rangle \diamond q$, can be found by determining the trees in Fig. 3, in subnetworks of $(\mathcal{N}', \mathcal{A}') = (\mathcal{N}_{\langle 1,4\rangle}, \mathcal{A}_{\langle 1,4\rangle})$. Computing $\mathcal{T}_4$ and $\mathcal{T}_5$ in $(\mathcal{N}', \mathcal{A}'(2)) = (\mathcal{N}, \mathcal{A})$, first two plots, and then analysing the nodes in $\mathcal{N}'$ and the arcs in $\mathcal{A}' - \mathcal{T}_s - \mathcal{T}_t$, two paths are found: $q_1 = \mathcal{T}_s(2) \diamond \mathcal{T}_t(2) = \langle 4,2,5\rangle$, of type 1, and $q_2 = \mathcal{T}_s(3) \diamond (3,5) = \langle 4,3,5\rangle$, of type 2. Either of them can be chosen since they both have the same cost, so it will be considered $p_1' = \langle 1,4,2,5\rangle$. In $(\mathcal{N}', \mathcal{A}'(5))$—trees represented in the last plots—$q_3 = \langle 4,3,5\rangle$ is the only path obtained. Thus $p_2' = \langle 1,4,3,5\rangle$. Finally, there are no paths besides p in $(\mathcal{N}', \mathcal{A}'(10))$; therefore, as $T(p_1') = 120$ and $T(p_2') = 90$, the path we are looking for is p_2'.
The method described above is summarized in Algorithm 1 and Procedures 1.1 and 1.2.

Algorithm 1. *Adaptation of algorithm of Katoh et al. to enumerate quickest simple paths*

```
$p_1 \leftarrow$ Quickest path from $s$ to $t$
$p \leftarrow$ Quickest path from $s$ to $t$, deviating from $p_1$ before $t$; $X \leftarrow \{p\}$
For ($i \in \{2,\dots,K\}$) Do
    $p_i \leftarrow$ Quickest path in $X$; $X \leftarrow X - \{p_i\}$
    $p_j \leftarrow$ Path analysed to obtain $p_i$
/*  Quickest path in $\mathcal{P}^j_{i+1}(v_\delta, v_{d(p_i)})$ */
    Delete $(v_\delta, x)$ such that $(v_\delta, x) \in \{p_j, \dots, p_{i-1}\}$
    $P_c \leftarrow$ Quickest path coincident with $p_j$ until $v_\delta$ and deviating before $v_{d(p_i)}$; $X \leftarrow X \cup \{P_c\}$
/*  Quickest path in $\mathcal{P}^j_{i+1}(v_{d(p_i)}, v_\gamma)$ */
    Delete $(v_\gamma, x)$ such that $(v_\gamma, x) \in \{p_j, \dots, p_{i-1}\}$
    $P_b \leftarrow$ Quickest path coincident with $p_j$ until $v_{d(p_i)}$ and deviating before $v_\gamma$; $X \leftarrow X \cup \{P_b\}$
```

```
/*  Quickest path in $\mathscr{P}^i_{i+1}(v_{d(p_i)+1}, t)$ */
    $P_a \leftarrow$ Quickest path coincident with $p_i$ until $v_{d(p_i)+1}$ and deviating after; $X \leftarrow X \cup \{P_a\}$
    Restore original network
EndFor
```

Procedure 1.1. *Adaptation of Katoh et al.'s procedure for the quickest path problem coincident with q until v_x and deviating before v_y*

```
Sort $\mathscr{A}$ by increasing order of the arcs capacities
$X \leftarrow \emptyset$
$i \leftarrow 1$
While ($i \leqslant r$) Do
    $\mathscr{N}_{\text{sub}_q(s,v_x)} \leftarrow \{u \in \mathscr{N} : u \notin \text{sub}_q(s, v_x) \vee u = v_x\}$
    $\mathscr{A}_{\text{sub}_q(s,v_x)}(c_i) \leftarrow \{(u,v) \in \mathscr{A} : u \in \mathscr{N}_{\text{sub}_q(s,v_x)} \wedge v \in \mathscr{N}_{\text{sub}_q(s,v_x)} \wedge c_{uv} \geqslant c_i\}$
    $p \leftarrow$ Minimum time path from $v_x$ to $t$, coincident with $q$ and deviating before $v_y$ in
    the modified network
    $X \leftarrow X \cup \{p\}$
    $j \leftarrow k$ such that $c(p) = c_k$;  $i \leftarrow j + 1$
EndWhile
$p^* \leftarrow$ Quickest path in $\{q \diamond p : p \in X\}$
```

Procedure 1.2. *Procedure for the minimum time path problem deviating from q before v_α*

```
Compute tree $\mathscr{T}_s$
If (($q$ is not defined) or ($q \neq \mathscr{T}_s(t)$)) Then
    $p \leftarrow \mathscr{T}_s(t)$
Else
    Compute tree $\mathscr{T}_t$
    $l^* \leftarrow +\infty$;  $X \leftarrow \{s\}$
    While ($X \neq \emptyset$) Do
        $u \leftarrow$ element in $X$;  $X \leftarrow X - \{u\}$
        If ($\xi_s(u) = \xi_t(u)$) Then
            For (($u,v$) $\in \mathscr{A} - \mathscr{T}_s - \mathscr{T}_t$ such that $\xi_s(u) < \xi_t(v)$) Do
                If ($l(\mathscr{T}_s(u)) + l_{uv} + l(\mathscr{T}_t(v)) < l^*$) Then
                    $l^* \leftarrow l(\mathscr{T}_s(u)) + l_{uv} + l(\mathscr{T}_t(v))$;  $u^* \leftarrow u$;  $v^* \leftarrow v$
                EndIf
            For (($u,v$) $\in \mathscr{T}_s$ such that $\xi_s(v) < \alpha$) Do $X \leftarrow X \cup \{v\}$
        EndIf
        If ($\xi_s(u) < \xi_t(u)$) Then
            If ($l(\mathscr{T}_s(u)) + l(\mathscr{T}_t(u)) < l^*$) Then
                $l^* \leftarrow l(\mathscr{T}_s(u)) + l(\mathscr{T}_t(u))$;  $u^* \leftarrow u$
            EndIf
            For (($u,v$) $\in \mathscr{T}_s$ such that $\xi_s(v) < \alpha$) Do $X \leftarrow X \cup \{v\}$
        EndIf
```

```
  EndWhile
  If (($u^*, v^*$) is defined) Then $p \leftarrow \mathscr{T}_s(u^*) \diamond (u^*, v^*) \diamond \mathscr{T}_t(v^*)$
  If ($u^*$ is defined) Then $p \leftarrow \mathscr{T}_s(u^*) \diamond \mathscr{T}_t(u^*)$
EndIf
```

To exemplify the algorithm presented here, consider the enumeration of $K = 3$ quickest paths in the network depicted in Fig. 1, with $\sigma = 100$. The set $\mathscr{A}$ will be considered to be sorted by the arc capacities: $c_1 = 2$, $c_2 = 5$ and $c_3 = 10$. First, a quickest path algorithm is used to determine $p_1 = \langle 1,3,4,5 \rangle$, with $T(p_1) = 35$. Then the quickest path deviating from p_1 before 5 is computed, thus looking for the minimum time path deviating from p_1 before 5 in each network $(\mathscr{N}, \mathscr{A}(c_i))$, $i = 1,2,3$ (see Procedures 1.1 and 1.2). Considering $q = \langle 1,3,4,5 \rangle$ and $v_y = 5$ in $(\mathscr{N}, \mathscr{A}(2)) = (\mathscr{N}, \mathscr{A})$, the path $p = \langle 1,2,5 \rangle = \mathscr{T}_s(5)$ with $c(p) = 2$ is obtained, since $q \neq \mathscr{T}_s(5)$. For network $(\mathscr{N}, \mathscr{A}(5))$, $q = \mathscr{T}_s(5) = \mathscr{T}_t(1)$ and $p = \langle 1,2,4,5 \rangle$. Finally $(\mathscr{N}, \mathscr{A}(10))$ is used, thus obtaining $p = \langle 1,4,5 \rangle$. From the three paths computed, the quickest, $\langle 1,2,4,5 \rangle$, is stored in X. The best element $p_2 = \langle 1,2,4,5 \rangle$ is selected in X and analysed. Node v_δ is not defined, since $v_{d(p_2)} = s = 1$, and $v_\gamma = t = 5$; therefore, only the quickest paths in $\mathscr{P}_3^1(1,5)$ and $\mathscr{P}_3^2(2,5)$ are computed and named, respectively, P_b and P_a. Considering the determination of $P_\mathrm{b}, (1,2)$ is deleted and the quickest path problem from 1 to 5, which deviates from p_1 before 5, is solved. In $(\mathscr{N}, \mathscr{A}(2))$, $p = \langle 1,3,5 \rangle$ is computed and $c(p) = 5$; then in $(\mathscr{N}, \mathscr{A}(10))$, $p = \langle 1,4,5 \rangle$. The quickest one is $P_\mathrm{b} = \langle 1,3,5 \rangle$, which is stored in X. After that P_a, the quickest path from 2 to 5, deviating from $\langle 2,4,5 \rangle$ before 5, is determined. Once again, networks $(\mathscr{N}, \mathscr{A}(2))$, where $\langle 2,5 \rangle$ is obtained, and $(\mathscr{N}, \mathscr{A}(5))$, determining $\langle 2,4,3,5 \rangle$, are considered. Thus, $P_\mathrm{a} = \langle 1,2,3,4,5 \rangle$ and $X = \{\langle 1,3,5 \rangle, \langle 1,2,3,4,5 \rangle\}$. The following quickest path is then selected in X, so $p_3 = \langle 1,3,5 \rangle$, $X = \{\langle 1,2,3,4,5 \rangle\}$, and the algorithm stops when K paths are found.

Regarding the number of operations performed, in the algorithm proposed K paths are selected in X and scanned. This analysis produces at most three new paths by applying Procedure 1.2 r times, in a worst case. That procedure demands the computation of two shortest paths trees, with $\mathscr{O}(m + n \log n)$, if Dijkstra's algorithm [11] is used, and the analysis of every arc and node in the network, with $\mathscr{O}(n + m)$. Therefore, it has computational complexity $\mathscr{O}(m + n \log n)$. Thus, the time complexity of the proposed algorithm is $\mathscr{O}(Kr(m + n \log n))$. In terms of memory requirements, at most $3K$ paths are generated, which have at most n nodes, so the space complexity is of $\mathscr{O}(Kn)$.

3. Computational experiments

In this section, experimental results regarding algorithms for ranking quickest paths are reported. The algorithms implemented were: Rosen et al.'s (`RA`); the one described in Section 2 (`NA`); Chen's, using Yen's algorithm to rank shortest paths (`CYA`), and Chen's, using Katoh et al.'s algorithm to rank shortest paths (`CKA`).

It should be noticed that, in order to solve this problem, Chen [8] proposed to rank K paths with minimum lead time in each $(\mathscr{N}, \mathscr{A}(c_i))$, $i = 1, \ldots, r$, storing all those paths and then selecting, by order, the K quickest ones. As referred to previously, to rank K shortest paths Yen's and Katoh et al.'s algorithms were used. Some of the paths may be obtained repeatedly. `CYA` and `CKA` store all the paths, and only in the ranking phase the repeated ones are ignored.

Table 2
Number of stored paths in undirected random networks with $\sigma = 1000$

	$d = 2$	$d = 10$
(a) $n = 1000$		
RA	698.875	617.100
NA	235.875	228.400
CYA	25909.500	48808.100
CKA	9231.750	19687.600
(b) $n = 5000$		
RA	851.800	732.400
NA	241.100	232.700
CYA	35339.200	50440.400
CKA	10270.100	19651.300

All the implementations were coded in C language and the tests were carried out on an AMD Athlon 1.3 GHz computer with 512 Mbytes of RAM. We present average results for solving 10 problems generated with the same parameters, except for a seed. In the tests presented $K = 100$ paths were ranked in undirected networks with both lead times and capacities uniformly generated in $\{1,\dots,100\}$, s and t are randomly chosen in $\{1,\dots,n\}$, and $\sigma = 1000$. Three types of experiments have been performed:

- Random connected network problems, with $n \in \{1000, 5000\}$ nodes, and m arcs (with $m = nd$ and $d \in \{2, 10\}$). A Hamiltonian path is created and then the remaining arcs are generated. There are no multiple arcs between a pair of nodes.
- Complete network problems, with $n \in \{100, 200, 500\}$ nodes, and arcs joining each pair of distinct nodes.
- Grid network problems, with $p = 100$ rows, $q \in \{50, 100, 150, 200\}$ columns and $n = pq$ nodes, arranged in a planar grid, numbered consecutively from left to right and top to bottom. Each pair of adjacent grid nodes is connected by an arc.

Table 2 shows the total number of paths generated and stored by each implementation until p_{100} is determined, while Table 3 presents the CPU times obtained. The results in the tables reflect the two different approaches of Rosen et al. and of Chen. In fact, in the first one, computing new paths is alternated with determining each p_k, while in the second all paths are generated before $p_1, \dots, p_K$ can be known. So, Chen's algorithm demands that more paths are computed and this is also shown, from an empirical point of view, in Table 2, and in Table 3 concerning running times.

In order to compute the performances of the algorithms until the last path is found, the following figures show partial results, concerning the number of paths generated and the CPU times, for obtaining the first $10, 20, \dots, 100$ paths. The results on those figures regard only RA and NA.

Those plots confirm what could already be expected from the theoretical complexity order and from Tables 2 and 3, that the algorithm based on Katoh et al.'s method, NA, in general, outperforms the one based on Yen's method, RA. In Tables 2 and 3 it can also be seen that CKA was more efficient than CYA, both in terms of time and memory space.

Table 3
Total CPU times (in s) in undirected random networks with $\sigma = 1000$

	$d = 2$	$d = 10$
(a) $n = 1000$		
RA	1.253	20.356
NA	0.901	7.183
CYA	11.863	135.564
CKA	8.140	51.119
(b) $n = 5000$		
RA	46.400	448.268
NA	17.833	159.266
CYA	47.025	618.996
CKA	54.997	537.571

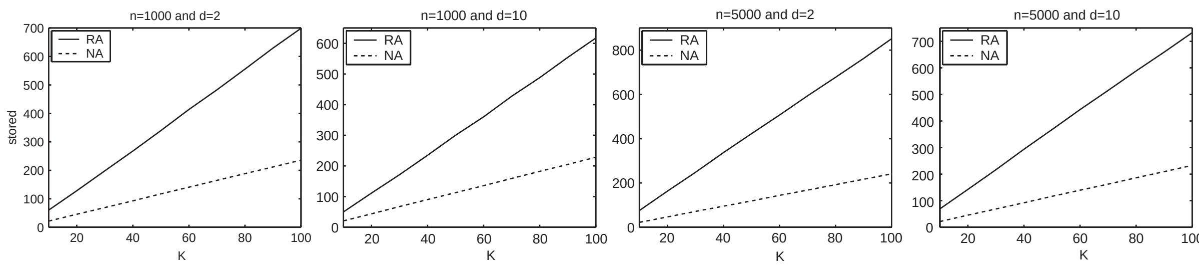


Fig. 4. Random undirected networks with $\sigma = 1000$.

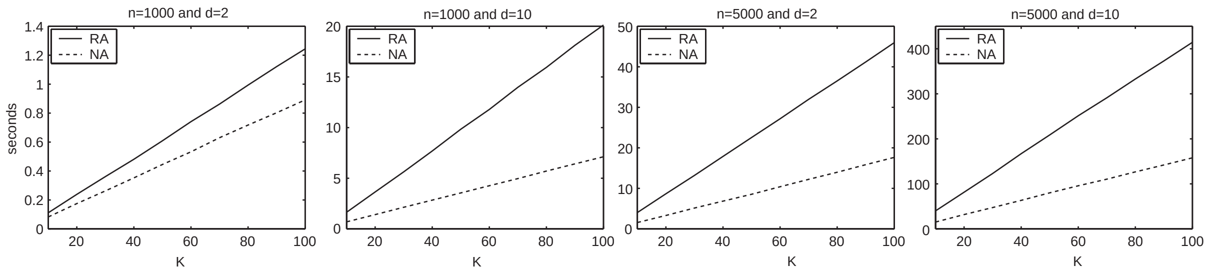


Fig. 5. Random undirected networks with $\sigma = 1000$.

Figs. 4 and 5 show random network results, while Figs. 6 and 7 concern complete networks, and Figs. 8 and 9 grid networks. Those plots show that NA outperformed RA and, as expected from the theoretical complexity results, both algorithms' CPU times increased linearly with K. Moreover, RA's behaviour seems to have a higher dependence on the size of the network. In fact, for any type of network in Figs. 5, 7 and 9 it is clear that RA CPU times increase faster than NA CPU times when n grows. This is quite evident in grid networks, where RA running times were better than the ones

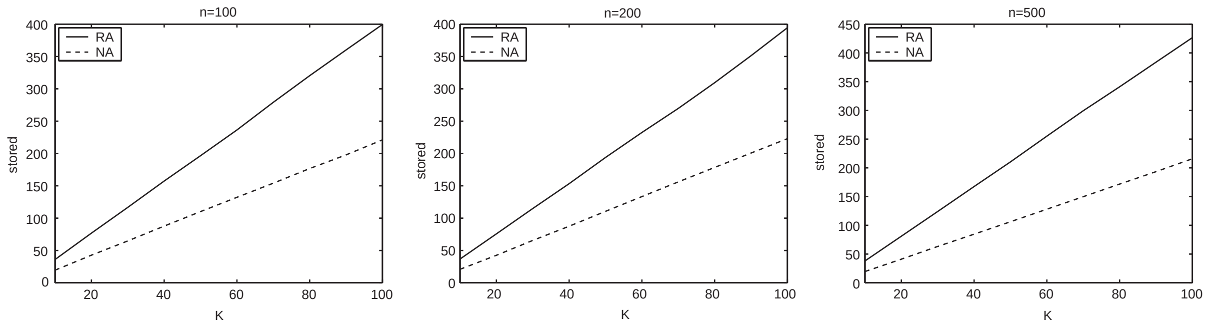


Fig. 6. Complete undirected networks with $\sigma = 1000$.

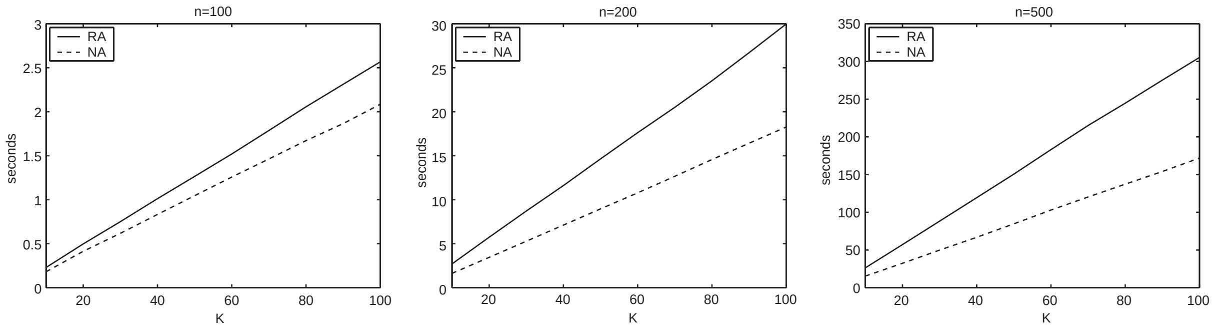


Fig. 7. Complete undirected networks with $\sigma = 1000$.

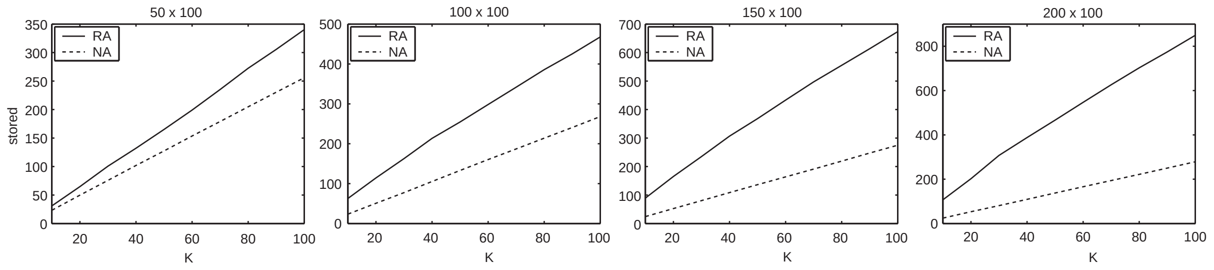


Fig. 8. Grid undirected networks with $\sigma = 1000$.

presented by NA for 50×100 grids, and they were very close for 100×100 grids, but the difference between them increased for 150×100 and 200×100 grids.

NA was also better than RA in terms of the number of stored paths. In fact, Figs. 4, 6 and 8 show that the number of stored paths was independent of the network size for NA, although it increased with K (recall that, at most, $3K$ paths are computed). On the other hand, the same number but concerning RA increased both with K and with n, being higher than for NA.

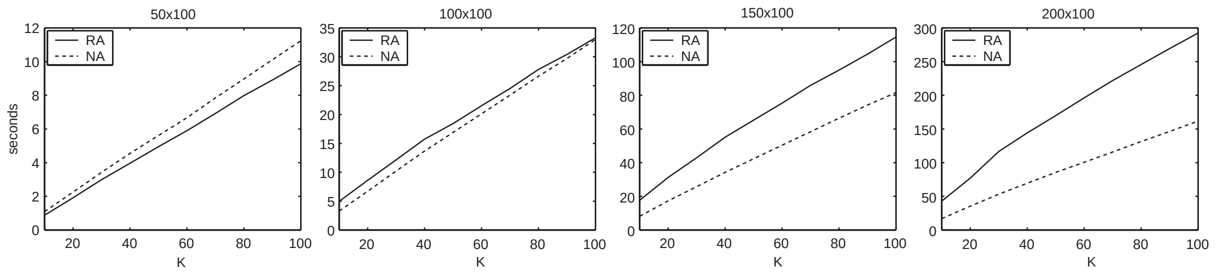


Fig. 9. Grid undirected networks with $\sigma = 1000$.

4. Conclusions

In this paper, a new algorithm for ranking quickest paths on undirected networks, which is an adaptation of Katoh et al.'s algorithm (of the same type of the adaptation of Yen's algorithm made by Rosen et al.), was presented. It was shown that this algorithm improves the time complexity of Rosen et al.'s algorithm, and the space complexity of Chen's algorithm. Experimental results confronted with Rosen et al.'s and Chen's algorithms results were presented. The new algorithm seems to be the most efficient.

Acknowledgements

The research of Marta Pascoal was developed within CISUC and partially supported by the Portuguese Ministry of Science and Technology (MCT), under the program PRAXIS XXI.

References

[1] Moore MH. On the fastest route for convoy-type traffic in flowrate-constrained networks. Transportation Science 1976;10:113–24.
[2] Chen YL, Chin YH. The quickest path problem. Computers & Operations Research 1990;17(2):153–61.
[3] Rosen JB, Sun SZ, Xue GL. Algorithms for the quickest path problem and the enumeration of quickest paths. Computers & Operations Research 1991;18(6):571–84.
[4] Martins EQV, Santos JLE. An algorithm for the quickest path problem. Operations Research Letters 1997;20: 195–8.
[5] Chen G-H, Hung Y-C. On the quickest path problem. Information Processing Letters 1993;46(3):125–8.
[6] Chen YL. An algorithm for finding the *K* quickest paths in a network. Computers & Operations Research 1993;20: 59–65.
[7] Martins EQV, Pascoal MMB. An algorithm for ranking optimal paths. Technical Report 01/004, CISUC, 2000 (`http://www.mat.uc.pt/~marta/Publicacoes/rank_optimal.ps.gz`).
[8] Chen YL. Finding the *K* quickest simple paths in a network. Information Processing Letters 1994;50:89–92.
[9] Yen JY. Finding the *K* shortest loopless paths in a network. Management Science 1971;17:712–6.
[10] Katoh N, Ibaraki T, Mine H. An efficient algorithm for *K* shortest simple paths. Networks 1982;12:411–27.
[11] Dijkstra E. A note on two problems in connection with graphs. Numerical Mathematics 1959;1:269–71.