Available online at www.sciencedirect.com

ScienceDirect

European Journal of Operational Research 181 (2007) 1045–1054

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Internet packet routing: Application of a *K*-quickest path algorithm

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

[a] *Faculdade de Economia da Universidade de Coimbra, Avenida Dias da Silva, 165, 3004-512 Coimbra, Portugal*
[b] *Instituto de Engenharia de Sistemas e Computadores—Coimbra, Rua Antero de Quental, 199, 3000-033 Coimbra, Portugal*
[c] *Departamento de Matemática, Polo I da Universidade de Coimbra, Apartado 3008, 3001-454 Coimbra, Portugal*
[d] *Departamento de Engenharia Electrotécnica e de Computadores, Polo II da Universidade de Coimbra, Pinhal de Marrocos, 3030-290 Coimbra, Portugal*
[e] *DEIO-CIO, Faculdade de Ciências, Universidade de Lisboa, Bloco C6, Campo Grande, 1749-016 Lisboa, Portugal*

Available online 3 May 2006

Abstract

This paper describes a study on the application of an algorithm to rank the *K*-quickest paths to the routing of data packets in Internet networks. For this purpose an experimental framework was developed by considering two types of random generated networks. To simulate values of the IP packet sizes, a truncated Pareto distribution was defined, having in mind to reflect a key feature of Internet traffic, namely its self-similar stochastic nature. Results concerning the average CPU times of the algorithm for the different sets of experiments will be presented and discussed.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Telecommunication networks; Quickest path; Simple paths ranking

1. Introduction

The quickest path problem is an optimal path problem where it is intended to compute a path along which a given amount of data can be sent with minimum transmission time, depending on its delay and on its bottleneck bandwidth (i.e., the minimum of its arc bandwidths). This problem has been studied by several authors from a network optimization point of view [2,5,10], and it has been extended to compute the K best routes, with $K > 1$ integer, by non-decreasing order of transmission times [1,7,10].

Conventional packet routing protocols (such as OSPF—open shortest path first—, or IS–IS—intermediate system–intermediate system) use shortest path algorithms, usually considering as (additive) metrics the

[*] Corresponding author. Address: Faculdade de Economia da Universidade de Coimbra, Avenida Dias da Silva, 165, 3004-512 Coimbra, Portugal. Tel.: +351 239 790595.
E-mail addresses: jclimaco@inescc.pt (J.C.N. Clímaco), marta@mat.uc.pt (M.M.B. Pascoal), jcrav@deec.uc.pt (J.M.F. Craveirinha), mecaptivo@fc.ul.pt (M.E.V. Captivo).

0377-2217/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2006.03.013

number of hops (see e.g. [14]) or the inverse of the available arc bandwidth. It is known that this approach is not very efficient in terms of network traffic carrying capacity and network resource utilization since it does not enable a balanced traffic distribution to be obtained in many working conditions. To improve the efficiency of packet routing algorithms it is important to consider multiple metrics such as paths bandwidth (minimal bandwidth among the arcs of the path) and delay, some of which are non-additive. Such approaches are associated with traffic engineering mechanisms of increasing importance in the new multiservice Internet technologies, namely multiprotocol label switching (MPLS). A central objective of those mechanisms is to adapt the routing decisions to variable network conditions, in terms of various quality of service (QoS) instances, by seeking to ‘‘optimise’’ the associated metrics; the ultimate aim is to obtain a distribution of the traffic flows across the network as favourable as possible, from the point of view of the multiple QoS requirements. In this model the objective function of the routing algorithm includes two metrics, delay and path bandwidth. The calculation of the *K*-quickest paths enables to consider implicitly various QoS dimensions as constraints. This procedure enables an efficient solution to the problem of eliminating paths not satisfying constraints regarding various QoS dimensions such as jitter or non-linear parameters such as path bandwidth.

In this work we consider the generation of alternative routes for packets in Internet, using the total transmission time as the metric function, and applying the *K*-quickest path algorithm introduced in [7]. This algorithm presented an efficient computational behaviour and is valid only in undirected networks. Note that there are many potential benefits of using multiple routes between a source and a destination. A known particular use of multiple routes is in ‘traffic splitting’ schemes where the offered load is divided by several routes having in mind to achieve a load balancing situation in the network. Another use of multiple routes is in alternative routing schemes of various kinds which enable to increase the robustness of the information transfer, associated with a given application, against packet losses resulting from network congestion or failure.

An experimental environment for application and test of the mentioned routing algorithm was implemented by defining a truncated Pareto distribution for simulating the packet sizes. This approximation has in mind to reflect a key feature of Internet traffic, namely its self-similar nature. The computational experiences were performed on two network topologies: random networks and networks based on the geographic coordinates of US cities.

This paper is divided into three parts: Section 2 describes the usage of the quickest path problem in Internet packet routing, Section 3 summarizes the *K*-quickest paths algorithm, and finally, Section 4 is devoted to the application model and computational experiments.

2. Internet packet routing

The routing of data packets in Internet has been traditionally based on shortest path algorithms using different types of additive metrics, such as arc (or link) count or delay. However, a number of studies has revealed that this simple procedure is not the most efficient approach from the point of view of various QoS criteria associated with end-to-end connections and overall network performance (see [11]). This has led to the interest in defining other more effective routing approaches, by including multiple path metrics, namely delay, bandwidth, link load (related to available bandwidth) or reliability, some of which are not additive. Concerning the application of this type of approaches one must be aware that the measurement or estimation of the parameters used in the objective function is a difficult task. This is a difficulty common to all routing approaches not based on hop account alone. Nevertheless, as stated above, it is known that the use of shortest path in terms of number of arcs it is not an efficient approach. Furthermore the efficiency of the algorithm makes it adequate for frequent recalculation of the paths in response to parameter changes.

A significant example is CISCOs enhanced interior gateway routing protocol (EIGRP), running in some Internet Provider networks, which uses a non-additive minimum bandwidth component by default [9]. Its standard objective function can be reduced to

$$T(p) = d(p) + \frac{\sigma}{B(p)},$$

for a path p. $T(p)$ represents the time to transmit σ data units throughout path $p, d(p) = \sum_{(i,j)\in p} d_{ij}$ is the total delay and $B(p) = \min_{(i,j)\in p}\{b_{ij}\}$ is the lowest bandwidth, both along p. The second term of function T gives the time spent to transmit the data in the slowest arc of the path. The consideration of this metric is equivalent to preferring paths which are shortest in terms of an additive combination of total packet delay and transmission time in the current slowest path link. In network optimization, the problem above,

$$\min\{T(p) : p \in \mathscr{P}\}, \tag{1}$$

where $\mathscr{P}$ denotes the set of feasible paths between an origin and a destination nodes, is known as the quickest path problem.

Although in communication networks problem (1) is often simplified and its resolution replaced by the computation of shortest–widest (that is, the shortest path among those with maximal bandwidth), or widest–shortest (that is, the maximal bandwidth path among the shortest paths) paths (see for instance [12,13]), specific algorithms for that problem have been presented. Chen and Chin presented the first algorithm known for this problem [2], and later Rosen et al. improved it [10] achieving an $\mathcal{O}(rm + rn \log n)$ time complexity and $\mathcal{O}(m + n)$ space complexity, where n, m and r denote the number of nodes, arcs, and distinct bandwidth values in the network. The algorithm [10] is based on finding the shortest path (with respect to d) in a sequence of subnetworks of $(\mathscr{N}, \mathscr{A})$, where the bandwidth lower-bound increases. Martins and Santos, also proposed an algorithm with the same worst-case complexity, after studying the problem as a bicriteria problem [5]. The result is very similar to Rosen et al.'s algorithm, but each shortest path is replaced by a maximum bandwidth path, among those with minimum delay, and it is sketched in Algorithm 1, where $(\mathscr{N}', \mathscr{A}'(w))$, $w > 0$, stands for the subnetwork of $(\mathscr{N}', \mathscr{A}')$ without arcs with bandwidth smaller than w.

Algorithm 1 (*Algorithm of Martins and Santos for the quickest path problem*)

```
$X \leftarrow \emptyset$
$p \leftarrow$ Widest–shortest path from $s$ to $t$ in $(\mathscr{N}, \mathscr{A})$
While ($p$ is defined) Do
   $X \leftarrow X \cup \{p\}$
   $p \leftarrow$ Widest–shortest path from $s$ to $t$ in$(\mathscr{N}, \mathscr{A}(B(p)))$
EndWhile
$p^* \leftarrow$ Quickest path in $X$
```

This study concerns the determination of more than one route for packets, i.e., the "K-best" routes, thus the algorithm proposed in [7] was applied, having in mind its efficiency. That algorithm is reviewed in the next section.

3. A review of the *K*-quickest path algorithm

One way to find "good" alternative paths is to compute the K-quickest paths by order of the transmission time, instead of only one, that is, given an integer K, to determine $p_1, \ldots, p_K$ such that:

- p_i is determined before p_{i+1}, for $i \in \{1, \ldots, K - 1\}$,
- $T(p_i) \leqslant T(p_{i+1})$, for $i \in \{1, \ldots, K - 1\}$,
- $T(p_K) \leqslant T(p)$, for $p \in \mathscr{P} - \{p_1, \ldots, p_K\}$,

which is known as the ranking quickest path problem, or the K-quickest paths problem. It will be assumed that these are loopless paths, that is, paths with no repeated nodes, but for the sake of simplicity only the term path will be used in the following. Two algorithms are known to solve that problem [1,10], and more recently another one was proposed [7] which presents a better computational performance. The latter will now be briefly reviewed and later it will be applied to an Internet packet routing model.

The algorithm uses a set X, initialized with the quickest path from s to t. Throughout the algorithm the best path is picked up from X, to be analysed, in order to generate new paths. Each path chosen in X is p_k, for some $k \in \{1, \ldots, K\}$, and the new paths are candidates to a future p_j, with $j > k$, therefore they are stored in X. The paths to be generated from each p_k are the quickest in a partition of path sets proposed by Katoh et al. [4].

Denote by $v_{d(p_k)}$ the deviation node of p_k, that is, the farthest node from s, where p_k deviates from $p_1,\ldots,p_{k-1}$, and let p_k be obtained as the best path in $\mathscr{P}^j_k(v_\delta, v_\gamma)$, where:

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

Given the paths p from s to i and q from i to t, $p \diamondsuit q$ denotes the path from s to t formed by p followed by q, and it is named the concatenation of p and q. $\mathscr{P}^j_k(v_\delta, v_\gamma)$ contains the paths $p = \mathrm{sub}_{p_j}(s, v_\delta) \diamondsuit q$, 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_γ. Katoh et al. noticed that the set of paths where p_k was computed can be partitioned according to

$$\mathscr{P}^j_k(v_\delta, v_\gamma) - \{p_k\} = \mathscr{P}^j_{k+1}(v_\delta, v_{d(p_k)}) \cup \mathscr{P}^j_{k+1}(v_{d(p_k)}, v_\gamma) \cup \mathscr{P}^k_{k+1}(v_{d(p_k)+1}, t)$$

and analysing p_k consists in computing the quickest path in each subset of this partition. Therefore the new paths that should be added to the set X of candidates are:

- the best one in $\mathscr{P}^j_{k+1}(v_\delta, v_{d(p_k)})$, i.e., that deviates from p_j between v_δ and $v_{d(p_k)}$,
- the best one in $\mathscr{P}^j_{k+1}(v_{d(p_k)}, v_\gamma)$, i.e., that deviates from p_j between $v_{d(p_k)}$ and v_γ,
- and the best one in $\mathscr{P}^k_{k+1}(v_{d(p_k)+1}, t)$, i.e., that deviates from p_k between $v_{d(p_k)+1}$ and t.

The resulting algorithm follows.

Algorithm 2 (*Algorithm to rank K-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 ($i \in \{2,\ldots,K\}$) 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 $\mathscr{P}^j_{i+1}(v_\delta, v_{d(p_i)})$ */
    Delete $(v_\delta, x)$ such that $(v_\delta, x) \in \{p_j,\ldots,p_{i-1}\}$
    $P_c \leftarrow$ 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 $\mathscr{P}^j_{i+1}(v_{d(p_i)}, v_\gamma)$ */
    Delete $(v_{d(p_i)}, x)$ such that $(v_{d(p_i)}, x) \in \{p_j,\ldots,p_i\}$
    $P_b \leftarrow$ 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
```

Consider now the determination of the quickest path in a set $\mathscr{P}^j_k(v_x, v_y)$. In order to avoid the calculation of repeated paths, some of its arcs cannot belong to q. The same happens with the nodes in $\mathrm{sub}_{p_j}(s, v_x)$ (except v_x), in order to obtain a path without loops. Let $(\mathscr{N}', \mathscr{A}')$ be the subnetwork of $(\mathscr{N}, \mathscr{A})$ without those arcs and nodes. Let also $b_1,\ldots,b_r$ be the distinct values of the arcs bandwidths.

A result by Pascoal et al. [7] states that:

Theorem 1. *Let $q_{j'}$ be the shortest path (in terms of delay) from v_x to t, deviating from* $\mathrm{sub}_{p_j}(v_x, t)$ *before $v_y \in p_j$, in $(\mathscr{N}', \mathscr{A}'(b_{j'}))$ and $p_{j'} = \mathrm{sub}_{p_k}(s, v_x) \diamondsuit q_{j'}$, with $j = 1,\ldots,r$. Then the quickest path in $\mathscr{P}^j_k(v_x, v_y)$ is p^* such that $T(p^*) = \min_{1 \leqslant j \leqslant r}\{T(p_{j'})\}$.*

Theorem 1 leads to the following method to find a quickest path in $\mathscr{P}^j_k(v_x, v_y)$.

Procedure 2.1. 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 bandwidth
$X \leftarrow \emptyset$
$i \leftarrow 1$
While $(i \leqslant r)$ Do
   $\mathscr{N}' \leftarrow \{u \in \mathscr{N} : u \notin \mathrm{sub}_q(s, v_x)\}$
   $\mathscr{A}'(b_i) \leftarrow \{(u,v) \in \mathscr{A} : u \in \mathscr{N}' \wedge v \in \mathscr{N}' \wedge b_{uv} \geqslant b_i\}$
   $p \leftarrow$ Minimum delay path from $v_x$ to $t$, coincident with $q$ and deviating before $v_y$ in $(\mathscr{N}', \mathscr{A}'(b_i))$
   $X \leftarrow X \cup \{p\}$
   $j \leftarrow k$ such that $B(p) = b_k; i \leftarrow j + 1$
EndWhile
$p^* \leftarrow$ Quickest path in $\{q \diamondsuit p : p \in X\}$
```

We will now focus on the determination of the shortest path q from s to t in a network $(\mathscr{N}, \mathscr{A})$, that deviates from $p' = \langle s = v_1, v_2, \ldots, v_{\ell(p')} = t\rangle$ before v_α, with $1 \leqslant \alpha < \ell(p')$. Let $\mathscr{T}_s$ and $\mathscr{T}_t$ be the trees of the shortest paths from s to any $i \in \mathscr{N}$, and from any $i \in \mathscr{N}$ to t, respectively. 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 $\xi_t(i)$). According to Katoh et al.,

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

Type 1. $\mathscr{T}_s(u) \diamondsuit \mathscr{T}_t(u)$, *with* $\xi_s(u) < \alpha$,
Type 2. $\mathscr{T}_s(u) \diamondsuit (u,v) \diamondsuit \mathscr{T}_t(v)$, *with* $(u,v) \in \mathscr{A} - (\mathscr{T}_s \cup \mathscr{T}_t)$ *and* $\xi_s(u) < \alpha$.

When the goal is to rank quickest paths we cannot assure that p' is the path with the minimal delay, therefore Lemma 1 cannot be applied straightforwardly. However, path q can be found according to the following scheme:

- If $p' \neq \mathscr{T}_s(t)$ then $q = \mathscr{T}_s(t) = \mathscr{T}_t(s)$ is the path to be used.
- Otherwise, $p' = \mathscr{T}_s(t) = \mathscr{T}_t(s)$, and Lemma 1 can be used, so q is the path of type 1 or type 2, above, which has minimal delay.

Procedure 2.2. Procedure for the minimum delay 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 $\mathscr{T}_t$
   $d^* \leftarrow +\infty; X \leftarrow \{s\}$
   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 $(d(\mathscr{T}_s(u)) + d_{uv} + d(\mathscr{T}_t(v)) < d^*)$ Then
               $d^* \leftarrow d(\mathscr{T}_s(u)) + d_{uv} + d(\mathscr{T}_t(v)); u^* \leftarrow u; v^* \leftarrow v$
            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 $(d(\mathscr{T}_s(u)) + d(\mathscr{T}_t(u)) < d^*)$ Then
            $d^* \leftarrow d(\mathscr{T}_s(u)) + d(\mathscr{T}_t(u)); u^* \leftarrow u$
         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 $p \leftarrow \mathscr{T}_s(u^*) \diamondsuit (u^*,v^*) \diamondsuit \mathscr{T}_t(v^*)$
   If $(u^*$ is defined) Then $p \leftarrow \mathscr{T}_s(u^*) \diamondsuit \mathscr{T}_t(u^*)$
EndIf
```

The worst-case computational time complexity of Algorithm 2 is $\mathcal{O}(Kr(m + n\log n))$, while its memory complexity is $\mathcal{O}(Kn)$. For full understanding of the algorithm [7] should be consulted.

4. Application model

An experimental environment for application and test of the mentioned routing algorithm was obtained by building the following model to simulate packet sizes and Internet routing conditions.

4.1. Simulation of Internet packet routing

A number of experimental based studies have shown that Internet traffic, unlike Markovian type traffic, possesses a specific property of self-similarity (see a review in [6, chapter 1]). This property is expressed by the fact that this traffic has similar or identical key statistical features in all time scales, namely concerning its correlational structure. In most cases of practical interest the self-similar nature of the Internet traffic is associated with a property of long-range dependence which means that the autocorrelation function $\rho(k)$ of the discrete stationary traffic process $\{X_t : t \in \mathbb{Z}\}$ decays slowly according to a hyperbolic function:

$$\rho(k) \sim ck^{-\beta}, \quad \text{as } k \to \infty, \tag{2}$$

where $0 < \beta < 1$ and $c > 0$. This implies that the autocorrelation function is not summable, unlike Markovian type processes which are said to be short-range dependent. In the case of exactly second-order self-similar processes with Hurst parameter $1/2 < H < 1$ which are characterized by an autocorrelation function of the form:

$$\rho(k) = \frac{1}{2}\left((k+1)^{2H} - 2k^{2H} + (k-1)^{2H}\right), \tag{3}$$

long-range dependence is verified (with $\beta = 2 - 2H$) and vice-versa. Similarly long-range dependence holds for asymptotically second-order self-similar processes, characterized by the fact that (3) is verified asymptotically, as $k \to \infty$. These traffic processes are designated as canonical self-similar models. These models also enable the representation of the typically bursty character of the traffic in a wide range of time scales.

There is a significant variety of stochastic models enabling the representation of the essential features of self-similar traffic, some of which seek the functional behaviour of the traffic sources to be reflected in the model (see an overview of simulation models for self-similar traffic in [3]). This type of models incorporate, in one form or another, specific distributions of sizes and durations of functional objects in the IP-networks (e.g. network protocols, data units, file sizes and TCP—transmission connection protocol—connections) which are known to be related to the long-range dependence property (see [6]). For this reason the distributions in question are heavy tailed, that is with asymptotical form

$$P(X > x) \sim cx^{-\alpha} \quad \text{as } x \to +\infty$$

with $c > 0$ and $0 < \alpha < 2$. Heavy tailed distributions have infinite variance for $1 < \alpha < 2$ and also an unbounded mean if $0 < \alpha \leqslant 1$.

A practical heavy-tailed distribution often used in this context is the Pareto distribution:

$$F_X(x) = P(X \leqslant x) = 1 - \left(\frac{b}{x}\right)^{\alpha}, \tag{4}$$

where $\alpha > 0$, $x \in [b, +\infty[$ and $b \geqslant 0$ is the location parameter. The parameter α varies in $]1, 2[$ in order to guarantee the statistical properties required by associated traffic processes with long range dependence.

In particular this distribution can be used to generate computationally traces of asymptotically self-similar traffic. In fact it can be proved that the number of simultaneous occupations in a M/G/∞ queue with Poisson arrivals and heavy-tailed distribution (with $1 < \alpha < 2$) of the service times is an asymptotically second-order self-similar process, with $H = \frac{3-\alpha}{2}$.

On the other hand the Internet Protocol (IP) datagram format has a maximal number of bytes equal to 65,535, including the IP header and the data content (cf. [14]). Considering a lower bound on the packet size equal to the minimum Ethernet II transmission unit (64 bytes) we obtained two practical requirements on the bounds for the packet size in our application study. Taking further into account the above theoretical considerations on self-similar traffic we defined a *truncated Pareto distribution* as an approximation to represent the simulated packet sizes σ in the range $[\sigma_{\min}, \sigma_{\max}]$:

$$\widehat{F}_\sigma(x) = P(\sigma \leqslant x) = \frac{1 - (b/x)^\alpha}{F_\sigma(M)}, \tag{5}$$

where $b = \sigma_{\min}$, $M = \sigma_{\max}$, F is the standard Pareto distribution and the mean of the distribution $\widehat{F}$ is

$$\bar{\sigma} = E(\sigma) = \frac{b\alpha}{\alpha - 1} \frac{1}{F_\sigma(M)}.$$

Note that the variance of the distribution (5) is finite (albeit with great value) depending on the values b and M, unlike the corresponding standard Pareto distribution which has infinite variance for $\alpha \in]1, 2[$. To generate random packet sizes according to (5) we can use the method of the inverse transform, based on the r.v. $\mathscr{U}$ uniformly distributed in $[0, 1]$:

$$x = \widehat{F}^{-1}(u) \Rightarrow x = b(1 - u\gamma)^{-1/\alpha}, \tag{6}$$

where $\gamma = F_\sigma(M)$ and u is a value of $\mathscr{U}$ obtained from any appropriate uniform random number generator. The bounds on the packet sizes used in the study are therefore $b = 512$ bits, and $M = 524280$ bits. As stated above the parameter α is related to the Hurst parameter $H = \frac{3-\alpha}{2}$ $(1/2 < H < 1)$ which is a measure of self-similarity of the associated traffic process described by the number of busy servers of the M/G/∞ queue with Pareto distributed service times. In the present application we have considered $\alpha = 1.2$ ($H = 0.9$) and $\alpha = 1.5$ ($H = 0.75$).

As for the delays experienced by the packets on the network links (transmission arcs), these were obtained from results in the large-scale measurement based study on packet dynamics [8], assuming a variation in the corresponding average number of arcs per path, from 1.5 to 7.5. This approximation leads to the empirical distribution in Table 1.

Concerning the values to be used as path bottleneck bandwidths $B(p)$ it should be noted that the estimation of such values based on measurements in a real network is very complex due to the extremely complicated functional working patterns of the Internet (see [8]). Taking as basis some histograms presented in [8] we have obtained an empirical distribution for the bottleneck bandwidths b_{ij} (in bits/millisecond), given in Table 2.

Note that this type of empirical distributions is quite variable, depending critically on the size of the analysed network, sites where the underlying measurements were taken, time periods of the measurements and estimation procedures. The empirical distributions in Tables 1 and 2 are just examples of possible situations, based on actual measurements in a large scale study, not "typical" average patterns.

Table 1
Arcs delay empirical distribution

d_{ij} (millisecond)	11	16	25	42	73	128	227	410	744	1365	2520	4681	8700
%	2.3	5.4	8.5	10.0	12.0	11.0	10.0	11.0	8.5	7.0	5.0	5.0	4.3

Table 2
Arcs bandwidth empirical distribution

b_{ij} (bits/millisecond)	1360	64	128	256	800	1680	2640	4000	8000
%	51.30	7.15	5.30	0.88	4.40	19.47	4.40	2.70	4.40

4.2. Application experiments

When Algorithm 2 was introduced by Pascoal et al.'s [7], extensive computational experiments were presented, with networks of several types and sizes, comparing it both with Rosen et al.'s and with Chen's algorithms. The two main conclusions drawn from this experiments were:

- The comparison between Chen's algorithm and the two others is not straightforward, once it separates into 2 phases the generation of paths and the selection of the *K*-quickest ones. This also implies the determination of more paths and, consequently, a higher total execution time.
- In general Pascoal et al.'s algorithm outperformed Rosen et al.'s. On the one hand the latter showed a higher dependence on the size of the network, on the other hand its CPU times increased faster than Pascoal et al.'s ones.

In the following some results of the application of Algorithm 2 to the determination of *K*-quickest paths, in the case of data packet routing in the Internet, will be reported.

In terms of the network topology, the algorithm was applied to two sets of networks. In the first set randomly generated networks with $m = 4n$ arcs, based on a rectangular grid of 400×240 (corresponding to a mesh size unit of 10Km), were obtained. Hence n nodes were randomly located in this grid structure, where $n \in \{500, 1000, \ldots, 3000\}$. It was imposed that each node is adjacent to at least 2 and at most 10 other nodes and all generated networks contain at least one Hamiltonian path. The second set of networks was obtained by using geographic coordinates of 1088 US cities (these coordinates were downloaded from the URL www.realestate3d.com/gps/latlong.htm). A node was assigned to each city and m arcs were randomly generated, guaranteeing an average node degree equal to 4 and a minimal (maximal) degree per node equal to 2 (10). In both sets undirected connected networks were considered, where each node has a degree between 2 and 10, and $m = 4n$ arcs.

For each size n, ten networks were generated, using ten different seeds. For each problem the $K = 50$ quickest paths between $\lfloor \frac{n^2}{2500} \rfloor$ source–destination pairs randomly chosen, were ranked by non-decreasing order of the transmission time, considering a different σ value for each source–destination pair. In possible applications of the algorithm in present teletraffic engineering mechanisms it is not necessary to select (for a given node-pair and application) more than a limited number of routes, say 5 or even less. Nevertheless the calculation of a larger number of *K*-quickest paths might be useful or even recommended as a first step of a route selection procedure. For example, this calculation could be used for generating candidate paths in a QoS routing model with an objective function of the given structure but including additional constraints related to other QoS metrics. Furthermore the experimentation with high values of *K* is useful for testing the efficiency of the algorithm, also having in mind other possible unexpectable applications. In those tests an implementation, `KQPA`, in C language of the *K*-quickest paths algorithm in Section 3, was used. The experience was carried out on an AMD Athlon 1.3 GHz computer with 512 MB of RAM. The results of the application of `KQPA` to the first set of experiments are summarized in Fig. 1, while Fig. 2 refers to the tests that used US-type networks.

It should be noted that the running times depend linearly on the number *K* of paths determined, and that it increases also with the size of the network. In fact, `KQPA` computed the 50 quickest paths in an average time of 0.103 seconds for networks with 500 nodes, and that time is of 1.743 seconds if $n = 3000$. It can also be noticed that the difference between the CPU times is larger when *K* increases. In terms of the path transmission times, they also grow with *K* and with n, varying from 342.059 to 391.194 milliseconds for p_{50}, in networks with 500 and 3000 nodes, respectively. In fact, when the network dimension is higher, the average number of arcs in each path also increases, and this results in an increase in the path transmission times.

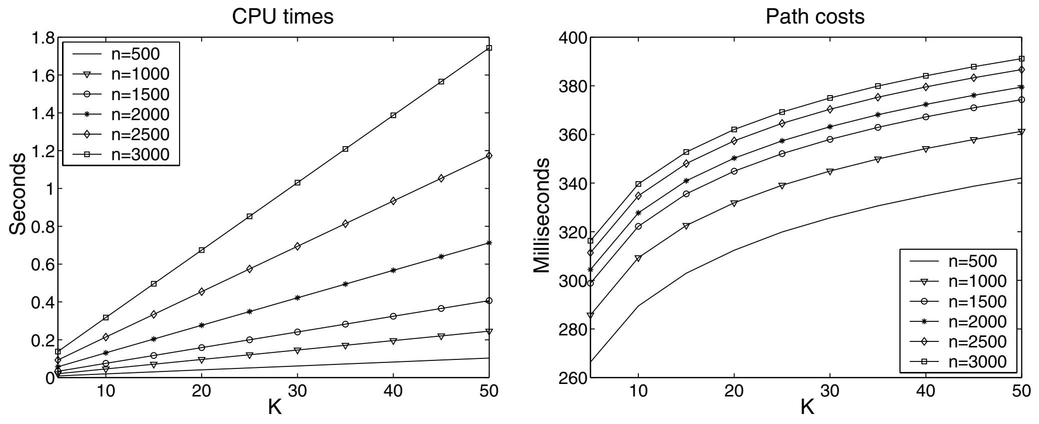

Fig. 1. Average results of KQPA on random networks, $K = 50$.

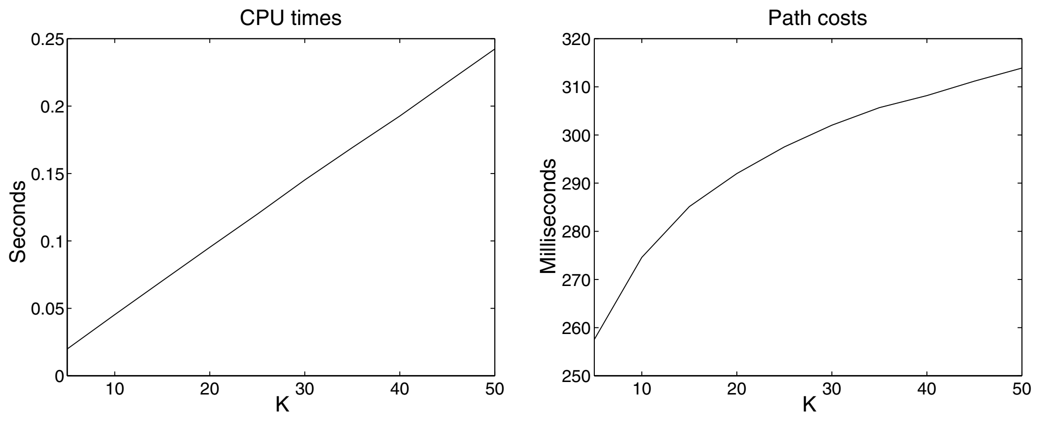

Fig. 2. Average results of KQPA on US-type networks, $K = 50$.

As expected, the results obtained in the networks based on the US cities coordinates follow the same trends as the ones obtained for the first set of experiments, with 1000 nodes random networks.

5. Conclusions

In this work we considered the generation of alternative routes for packets in Internet, using the total transmission time as the metric function and applying an efficient K-quickest path algorithm. A probabilistic model was constructed to generate packet sizes, using a truncated Pareto distribution and to simulate Internet routing conditions. The computational experience was performed on the two types of network topologies: random generated networks and networks based on the geographic coordinates of US cities. It must be emphasized that the running times depend linearly on the number K of determined paths and that it also increases with the size of the network. In any case, it can be concluded that the procedure is efficient in practical situations. In the worst tested situation the 50 best solutions in a network with 3000 nodes were obtained in 1.743 seconds.

As a major conclusion, the proposed approach is efficient enough to justify in many cases the use of the formulation (1) for the Internet packet routing problem (as permitted, for example, by the CISCOs routing protocol implementation EIGRP) instead of currently proposed simplifications such as the computation of shortest–widest or widest–shortest paths.

Acknowledgments

This research was partially supported by POSI/SRI/37346/2001 (Project title: "Modelos e Algoritmos para Tratamento da Incerteza em Sistemas de Apoio à Decisão"), POCTI/MAT/139/2001 and POCTI/ISFL-1/152.

References

[1] Y.L. Chen, Finding the *K* quickest simple paths in a network, Information Processing Letters 50 (1994) 89–92.

[2] Y.L. Chen, Y.H. Chin, The quickest path problem, Computers and Operations Research 17 (2) (1990) 153–161.

[3] R. Girão-Silva, J. Craveirinha, Study on simulation models for self-similar traffic, in: Proceedings of the 6-th IEEE International Conference High-Speed Networks and Multimedia Communications, 2003, Springer-Verlag, 2003, pp. 571–580.

[4] N. Katoh, T. Ibaraki, H. Mine, An efficient algorithm for *K* shortest simple paths, Networks 12 (1982) 411–427.

[5] E.Q.V. Martins, J.L.E. Santos, An algorithm for the quickest path problem, Operations Research Letters 20 (1997) 195–198.

[6] K. Park, W. Willinger (Eds.), Self-similar Network Traffic and Performance Evaluation, Self-similar Network Traffic: An Overview, Wiley-Interscience, 2000, pp. 1–38 (Chapter 1).

[7] M.M.B. Pascoal, M.E.V. Captivo, J.C.N. Clímaco, An algorithm for ranking quickest simple paths, Computers and Operations Research 32 (3) (2005) 509–520.

[8] V. Paxson, End-to-end Internet packet dynamics, IEEE/ACM Transactions on Networking 7 (3) (1999) 277–292. Available from: <http://citeseer.nj.nec.com/paxson97endtoend.html> .

[9] A. Riedl, D.A. Schupke, A flow-based approach for IP traffic engineering utilizing routing protocols with multiple metric types, in: Proceedings of the 6th INFORMS Telecommunication Conference, Boca Raton, USA, March 2002. Available from: <http://www.informs.org/Conf/Telecom02/Abstracts/Riedl01351033926.pdf>.

[10] J.B. Rosen, S.Z. Sun, G.L. Xue, Algorithms for the quickest path problem and the enumeration of quickest paths, Computers and Operations Research 18 (6) (1991) 571–584.

[11] S. Savage, A. Collins, E. Hoffman, J. Snell, T.E. Anderson, The end-to-end effects of Internet path selection, in: Proceedings of ACM SIGCOMM'99, September 1999, pp 289–299. Available from: <http://citeseer.nj.nec.com/savage99endtoend.html>.

[12] J.L. Sobrinho, Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet, in: Proceedings of the IEEE Infocom 2001 Conference, Anchorage, Alaska USA, April 2001, pp 727–735. Available from: <http://www.ieee-infocom.org/2001/paper/105.pdf>.

[13] J.L. Sobrinho, Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet, IEEE Transactions on Networking (August) (2002) 541–550.

[14] F. Wilder, A Guide to the TCP/IP Protocol Suite, Artech House, Inc., Boston, 1998.