An Algorithm for Calculating the K Most Reliable Disjoint Paths with a Maximum Number of Arcs

by

Teresa Gomes[1,2]

José Craveirinha[1,2]

Lúcia Martins[1,2]

Marta Pascoal[3,4]

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

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

[3]Centro de Informática e Sistemas da Universidade de Coimbra

[4]Departamento de Matemática
Pólo I da Universidade de Coimbra,
Largo de D. Dinis, 3000 COIMBRA, Portugal.

INESC – Coimbra

Research Report ET-N11
January 2001

Work 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 the K Most Reliable Disjoint Paths with a Maximum Number of Arcs

Teresa Gomes[(1,2)], José Craveirinha[(1,2)], Lúcia Martins[(1,2)],
Marta Pascoal[(3,4)]

[(1)]Departamento de Engenharia Electrotécnica,
Pólo II da Universidade de Coimbra,
Pinhal de Marrocos, 3030 COIMBRA, Portugal.

[(2)]INESC-Coimbra, Rua Antero de Quental 199,
3000-033 COIMBRA, Portugal.

[(3)]Centro de Informática e Sistemas da Universidade de Coimbra
[(4)]Departamento de Matemática
Pólo I da Universidade de Coimbra,
Largo de D. Dinis, 3000 COIMBRA, Portugal.

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

Abstract

Telecommunications networks are expected to achieve specified levels of survivability under various failure modes and conditions. Due to the extended use of optical fibers, every link in a transmission network carries a significant amount of traffic.

A main concern is the network ability to allow communication between all its operational nodes in the presence of failures. Network reliability and survivability are improved if node/link disjoint paths are used between source and terminal nodes in the network. Also in telecommunications networks the problem of calculating paths between pairs of nodes involves, in general, constraints on the maximum number of arcs in any path.

Two very efficient algorithms will be presented that allows the enumeration of the most reliable s-t disjoint paths with a maximum number of arcs, in a network where links and/or nodes fail independently, with known probability.

The number of obtained paths can result from of a pre-specified end-to-end reliability, whenever this may be achieved, or by a set of k most reliable disjoint paths.

Keywords: Network reliability, network survivability, disjoint shortest paths enumeration.
Topics: Reliability in communications networks.

1 Introduction

Telecommunications networks are expected to achieve specified levels of survivability under various failure modes and conditions [KDP95]. Due to the extended use of optical fibers, every link in a transmission network carries a significant amount of traffic. For this reason several

automatic procedures for network recovery are already part of SDH (Synchronous Digital Hierarchy)/SONET (Synchronous Optical NETworks) networks. Also the use of SDH/SONETS Digital Cross-connects Systems (DCS's) allows re-routing of calls in case of link or node failures [DML94, KDP95, San96].

A main concern is the network ability to allow communication between all its operational nodes in the presence of failures. Network reliability and survivability are improved if node/link disjoint paths are used between source and terminal nodes in the network.

It is assumed that the links and nodes of a network under study have known reliability and that the cost of a path between any node pair, s-t, is its probability of being operational. Disjoint paths will then be selected by decreasing order of their reliability which will ensure we are using an adequate set of disjoint paths from a reliability point of view. In a network where both links and nodes can fail independently, the selected paths should be node disjoint. In networks where only links fail independently, it is sufficient if the selected paths are link disjoint.

Algorithm for both cases (networks where only links can fail and networks where links and nodes fail) will be presented which are based on an extremely efficient algorithm enabling to enumerate the K shortest paths with a maximum number of arcs in [GMC01] derived from the loopless version of the algorithm proposed in [MPS99a].

The report is organised as follows. Firstly the motivation for calculating the set of K-most reliable disjoint paths with a maximum number of arcs in a reliability analysis context, will be presented. Next the foundations of the algorithms are presented namely by showing the way in which the formulated problem can be transformed into a K-shortest disjoint paths constrained problem. After reviewing the mathematical concepts underlying basic steps of the algorithm, this is be formalised. In section 5 computational examples of application of the algorithm to networks of large dimension are presented in order to illustrate its efficiency and applicability.

2 Motivation

There are several advantages in seeking two or more disjoint paths between a source and terminal nodes, as Torrieri in [Tor92] points out:

- if one or more paths fail, other paths are available, therefore increasing network reliability and survivability;
- re-routing is simplified in the case of node or link failure, because alternate paths will be operational with great probability – they don't share resources with the unavailable path;
- when network-wide time delay is to be minimised multiple-path routing usually provides optimal solution; these paths are often disjoint;
- when a path fails and a new one is selected, there is no problem of possible occurrence of loops.

Also in telecommunications networks the problem of calculating paths between pairs of nodes involves, in general, constraints on the maximum number of arcs in any path. This limitation usually results from maximum a allowed propagation delay (at transmission level) or from routing constraints in teletraffic networks (maximum number of hops).

In this paper algorithms will be proposed for selecting disjoint paths by decreasing order of probability. The number of selected paths will usually result from of a pre-specified end-to-end reliability, whenever they may be achieved.

Suurballe in [Suu74] proposed efficient algorithms for obtaining the K-shortest (node or links) disjoint paths. Suurballe basically transforms the problem into a minimum cost flow problem [AMO93], which can be solved, for example determining the K successive shortest incremental chains. Torrieri [Tor92] proposed an algorithm for obtaining an optimal set of short disjoint paths, with a maximum number of arcs, in a communications network assuming that link costs are uniform. The set of paths in [Tor92] is optimal in the sense that it contains the shortest disjoint paths which minimise the number of subsequent exclusions of shortest paths in posterior selections.

Suurballe's algorithm [Suu74] can not be adapted to the problem under analysis because it does not allow to limit the number of arcs used in each path.

Also our approach is different from Torrieri [Tor92] since we assume that the links and nodes of the network have known reliability and the cost of the path is not given by the number of links but by its probability of being operational.

The fact that disjoint paths are selected by decreasing order of their reliability will ensure we are using an adequate set of disjoint paths from a reliability point of view. The proposed algorithm is very efficient because it is based on an extremely fast algorithm [MPS99b], for obtaining the K-shortest paths between any given pair of nodes. Also an extremely efficient algorithm enabling to enumerate the K shortest paths with a maximum number of arcs, based on the loopless version of the former algorithm, [MPS99a], was presented in [GMC01].

2.1 Context of application

The motivation for selecting disjoint paths was already given at the beginning of this section 2. We now will give some examples of applicability of this approach.

In the context of network reliability analysis, given the failure probabilities of nodes and links in a telecommunications network, the proposed algorithms identifies a set of (alternative) most reliable paths which assures a given end-to-end reliability (if possible). This set can be used to decide where extra transmission equipment should be installed for assuring uninterrupted communication between nodes of present day transmission networks, where very hight rates are used between node pairs and any disruption can have very heavy penalties for the service provider.

As Ball in [Bal79] points out, certain measures of network reliability arise as part of the problem of network design. Given the failure probabilities of nodes and link, the problem

of obtaining a minimum cost network is a very difficult problem. In this type of problem, reliability measures are often repeatedly calculated until an adequate network is obtained; a possible measure of network adequacy from a reliability point of view could be a minimum number of disjoint paths and/or a given minimum end-to-end reliability obtained using only disjoint paths.

As Ball [Bal79] points out, certain measures of network reliability arise as part of the problem of network design. Given the failure probabilities of nodes and links, the problem of obtaining a minimum cost network is a very difficult problem. In this type of problem, path reliability related measures are often repeatedly calculated until an adequate network is obtained. A possible measure of network adequacy from a reliability point of view could be a minimum number of disjoint paths or, more elaborate, a given minimum end-to-end reliability obtained using only disjoint paths, as it's proposed here.

In teletraffic networks, and in the context of dynamic routing, one of the criteria for selecting a path between a given pair of nodes could be its probability of being operational, its blocking probability and the fact of being disjoint with previously used ones. The fact that the new selected path is disjoint with previously selected paths, will increase the probability that this new path will not be suffering from the same problems which affected previously used paths.

3 Foundations of the algorithms

Let $G = (N, L)$ be a directed graph where $N = \left\{v_1, v_2, \ldots, v_{|N|}\right\}$ is the node set and L the arc (or link) set, composed of ordered pairs of elements in N, where $|N|$ represents the cardinality of set N. If $l = (i, j)$, j is the head of l and i its tail. A path from s to t ($s, t \in N$) in this graph will be specified by the sequence $p = \langle s, (s, v_1), v_1, \ldots, (v_w, t), t\rangle$, where all $l = (v, u) \in p$ belong to L. If all nodes in p are different it is called a loopless path. Although up till now only the term path was used, the adjective loopless was implicitly considered. The word "loopless" will continue to be omitted until explicit reference is need.

Each node $v \in V$ (link $l \in L$) has an operational probability $p_V(v)$ ($p_L(l)$). In a network where nodes and links fail (independently), and (node) disjoint paths from s to t are seeked, a cost matrix $[c_{ij}]$ of dimension $|N| \times |N|$ is defined such that the cost of an arc is the additional cost of introducing that arc and its head in a path:

$$c_{ij} = \begin{cases} -\ln\left(p_V(j)p_L(l)\right) & \text{if } l = (i,j) \in L \\ +\infty & \text{if } l = (i,j) \notin L \end{cases} \tag{1}$$

The cost of a path $p = \langle s, (s, v_1), v_1, \ldots, (v_w, t), t\rangle$ is:

$$\mathcal{C}(p|s) = \sum_{(v_i, v_j) \in p} c_{v_i v_j} \tag{2}$$

and its reliability is:

$$\Pr(p) = p_V(s) e^{-\mathcal{C}(p|s)} \tag{3}$$

In equation (2) we write $\mathcal{C}(p|s)$ instead of simply $\mathcal{C}(p)$ because the cost of the paths selected by the k-shortest path algorithm are obtained considering that s doesn't fail; the failure probability of s is taken into consideration when $\Pr(p)$ is calculated, as shown in equation (3). Note that if $\mathcal{C}(p_1|s) < \mathcal{C}(p_2|s)$ then $\Pr(p_1) > \Pr(p_2)$, so the order of the paths is not changed due to the constant factor $p_V(s) > 0$. The idea of associating to an arc the operational probability of its head is already present in [Tor94].

Note that if network nodes don't fail, $p_V(v) = 1$ $(v = 1, \ldots, |N|)$, the definition of cost matrix $[c_{ij}]$ and equation (3) are still valid – although unecessarily elaborate.

Equation (3) establishes a relation between the cost of a path and its reliability. Using the cost matrix $[c_{ij}]$, the enumeration of the k shortest paths with a maximum of D arcs is equivalent to enumerating, by decreasing order of their probability, the k most reliable paths in a network with at most D arcs.

In the algorithms, presented in the next section, (link or node) disjoint paths can be enumerated until there are no more paths or a given reliability, R_{st}, for node-pair (s-t) is attained. Let $S_K = \{p_1, p_2, \ldots, p_K\}$ be the set of K paths used to connect s and t, and let $r_{st}(S_K)$ be the s-t reliability that can be attained if the set S_K is used to connect s-t (that is the probability of s-t being connected by any of the operational paths in S_K). Then $r_{st}(S_k)$, $k = 1, \ldots, K$ can be efficiently obtained, using any of the algorithms in the litterature for calculating the probability of a union of events.

4 Proposed Algorithms

KD [GMC01] is used as basis for the proposed algorithm. Note that MPS and KD are both deviation algorithms. Each time a path p is chosen from a set of candidate paths, X, new paths may be added to X. In the context of the MPS algorithm the node v_k of path p, from which a new candidate path is generated is the *deviation node* of that new path; in a path the link the tail of which is the deviation node, is called the *deviation arc* of that path. By definition s is the deviation node of p_1 (the shortest path from s to t).

For enumerating node disjoint paths, in a network where arcs and nodes may fail, every time a path is selected all its nodes are marked as *used*; so when a path is chosen from X, a new constraint has to be satisfied: a path p will only be included in the set of disjoint paths if none of its nodes (except its end nodes s and t) is marked used.

Similarly, in a network where only links fail, every time a path is selected all its arcs are marked *used*; so when a path is chosen from X, a new constraint has to be satisfied: a path p will only be included in the set of link disjoint paths if none of its arcs is marked used.

4.1 Notation

The *concatenation* of paths p, from i to j, and q, from j to l, is the path $p \diamond q$, from i to l, which coincides with p from i to j and with q from j to l.

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, from i to t, in $\mathcal{T}_t$; the *reduced cost* $\bar{c}_{ij}$ of arc $(i,j) \in L$ associated with $\mathcal{T}_t$ is $\bar{c}_{ij} = \pi_j(\mathcal{T}_t) - \pi_i(\mathcal{T}_t) + c_{ij}$. So all arcs in $\mathcal{T}_t$ have a null reduced cost. The advantage of using reduced costs was first noted by Eppstein [Epp98] and is shown, by theorems 8 and 9 in [MPS99b] and by theorem 2.1 in [MPS99a], in the context of the MPS algorithm, to lead to less arithmetic operations and to sub-path generation simplification.

Let $\mathcal{T}_t^*$ be the tree of the shortest paths from all nodes to t; $p^*_{v_j t}$ represents the shortest path from v_j to t in $\mathcal{T}_t^*$. The sub-path from v_k to t in p is represented by $p_{v_k t}$, and the sub-path from s to v_k by $p_{s v_k}$.

Next additional notation and definitions used in the proposed algorithm and also in KD [GMC01], 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 of p 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 the set of arcs L of (V, L) be written in terms of $A(v)$, the set of arcs the tail node of which is $v \in V = \{1, \ldots, n\}$, *i.e.* $L = A(1) \cup A(2) \cup \ldots \cup A(n)$ such that $A(i) \cap A(j) = \emptyset$ 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 L 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 $L = \{a_1, \ldots, a_m\}$ is said to be in the *sorted forward star form*, which leads to the path generation simplification – for more details see [DGKK79].

4.2 Link disjoint paths algorithm

Let $disj_L(p)$ be a function which returns the value true if no arc in p is marked used. Let $d_L(p)$ be a function which returns the depth of the tail of the first used arc in p. If no arc in p, is used, it returns $|p|$.

Algorithm KD-ld

1. **Input**: (V, L), $p_L(a)$ with $a \in L$ $(\equiv c_{ij})$, D, P_d (target reliability), K, s and t.
2. **Output**: $S_z = \{p_1, p_2, \cdots, p_z\}$; $P_c = r_{st}(S_z)$.
3. By using Dijkstra inverse algorithm, obtain the shortest paths from every node to t, $\mathcal{T}_t^*$.
4. Calculate the reduced cost $\bar{c}_{ij}$ for every $(i,j) \in L$.
5. Rearrange the arcs of (V, L) in the sorted forward star form (for the computed $\bar{c}_{ij}$, leading to $L = \{a_1, \ldots, a_m\}$ such that for any $h \in \{1, \ldots, m-1\}$, $\bar{c}_{a_h} \leq \bar{c}_{a_{h+1}}$ if

$v = \text{tail}(a_h) = \text{tail}(a_{a_{h+1}}))$

6. $p \leftarrow$ shortest path from s to t ($p \in \mathcal{T}_t^*$).

7. $k \leftarrow 0$; $P_c \leftarrow 0$; $S_0 = \emptyset$.

8. $X \leftarrow \{p\}$ (X is the set of paths that are candidates to shortest path).

9. `While` $(k < K)$ and $(P_c < P_d)$ and $(X \neq \emptyset)$ `do`

   (a) $p \leftarrow$ shortest path in X

   (b) $X \leftarrow X - \{p\}$

   (c) `If` $(|p| \leq D)$ and (p has no loops) and $disj_L(p)$ `then`

      i. $k \leftarrow k + 1$

      ii. $p_k \leftarrow p$; $S_k \leftarrow S_{k-1} \cup \{p_k\}$

      iii. mark all arcs of p as $used$

      iv. Update P_c ($P_c \leftarrow r_{st}(S_k)$)

   `EndIf`

   (d) Let v be the deviation node of p

   (e) `If` $(|p| = D)$ and $(|p_{sv}| < D - 1)$
   `then` $l \leftarrow D - 2$ (try to generate new paths)
   `else If` $(|p| \neq D$ and $|p_{sv}| < D)$
   `then` $l \leftarrow \min(|p| - 1, D - 1)$ (try to generate new paths)
   `else` $l \leftarrow -1$ (no path will be generated from p)
   EndIf EndIf

   (f) $l \leftarrow \min(d_L(p), l)$

   (g) `If` $l \neq -1$ `then` (possibly a path will be placed in X)

      i. $v_i \leftarrow v$

      ii. `Repeat`

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

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

         C. `If` v_i is the tail of a_{h+1} `then`
         - $v_j \leftarrow$ head of a_{h+1}
         - $X \leftarrow X \cup \{p_{sv_i} \diamond \langle v_i, a_{h+1}, v_j \rangle \diamond p^*_{v_j t}\}$

         `EndIf`

         D. `If` $d_p(v_i) = l$
         `then` $l \leftarrow -1$ (no more paths will be obtained from p)
         `else` $v_i \leftarrow$ following node in ${}^l p$
         `EndIf`

      `Until` (p_{sv_i} has a loop) or $(l = -1)$

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

`EndWhile`

Now some comments on conditions that affect the flow of instructions of KD-nd, relative to algorithm KD, will be made.

Obviously step 9c has only the additional constraint, $disj_L(p)$, that no arc of p is used. Step 9(g)iiB of the algorithm assures that only potentially selectable disjoint paths (or paths that can possibly originate potentially selectable disjoint paths) with at most D arcs paths are added to X. The condition, "(a_{h+1} is marked *used*)", not present in algorithm KD, establishes that new paths will only be added to the set of candidate paths if all arcs up to (and including) its deviation arc are not used. This guarantees that a path added to X is disjoint with previously selected paths or/and can possibly originate new paths which are disjoint with the ones already identified. Finally, variable l in KD [GMC01] gives the maximum depth of the nodes that can be deviation node of new paths – for the purpose of generating paths with at most D arcs; step 9f in KD-nd redefines this depth taking into consideration that the head of an arc marked used cannot be a deviation node.

4.3 Node disjoint paths algorithm

Let $disj_V(p)$ be a function which returns the value true if no node in p (except s and t) is marked used. Let $d_u(p)$ be a function which returns the depth of the first used node in p, after s. If no node in p, after s, is used, it returns $|p| - 1$.

Algorithm KD-nd

1. **Input**: (V, L), $p_L(a)$ with $a \in L$, $p_V(v)$ with $v \in V$, D, P_d (target reliability), K, s and t.
2. **Output**: $S_z = \{p_1, p_2, \cdots, p_z\}$; $P_c = r_{st}(S_z)$.
3. Calculate c_{ij}.
4. By using Dijkstra inverse algorithm, obtain the shortest paths from every node to t, $\mathcal{T}_t^*$.
5. Calculate the reduced cost $\bar{c}_{ij}$ for every $(i, j) \in L$.
6. Rearrange the arcs of (V, L) in the sorted forward star form (for the computed $\bar{c}_{ij}$ leading to $L = \{a_1, \ldots, a_m\}$, such that for any $h \in \{1, \ldots, m-1\}$, $\bar{c}_{a_h} \leq \bar{c}_{a_{h+1}}$) if tail($a_h$)=tail($a_{h+1}$)
7. $k \leftarrow 0$; $P_c \leftarrow 0$; $S_0 = \emptyset$; mark nodes s and t as *used* ;
8. $X \leftarrow \{p_{st}^*\}$ (X is the set of paths that are candidates to shortest path).
9. `While` ($k < K$) and ($P_c < P_d$) and ($X \neq \emptyset$) `do`
   (a) $p \leftarrow$ shortest path in X
   (b) $X \leftarrow X - \{p\}$
   (c) `If` ($|p| \leq D$) and (p has no loops) and $disj_V(p)$ `then`
      i. $k \leftarrow k + 1$
      ii. $p_k \leftarrow p$; $S_k \leftarrow S_{k-1} \cup \{p_k\}$
      iii. mark all nodes of p as *used*

iv. Update P_c ($P_c \leftarrow r_{st}(S_k)$)

`EndIf`

(d) Let v be the deviation node of p

(e) `If` $d_u(p) > d_p(v)$
`then`

i. `If` ($|p| = D$) and ($|p_{sv}| < D - 1$)
`then` $l \leftarrow D - 2$ (try to generate new paths)
`else If` ($|p| \neq D$ and $|p_{sv}| < D$)
`then` $l \leftarrow \min(|p| - 1, D - 1)$ (try to generate new paths)
`else` $l \leftarrow -1$ (no path will be generated from p)
`EndIf`
`EndIf`

ii. $l \leftarrow \min(d_u(p) - 1, l)$ (maximum depth of potential deviation nodes)

`else` $l \leftarrow -1$
`EndIf`

(f) `If` $l \neq -1$ `then` (possibly path(s) will be placed in X)

i. $v_i \leftarrow v$

ii. `Repeat`

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

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

C. `If` v_i is the tail of a_{h+1} `then`
- $v_j \leftarrow$ head of a_{h+1}
- $X \leftarrow X \cup \{p_{sv_i} \diamond \langle v_i, a_{h+1}, v_j \rangle \diamond p^*_{v_j t}\}$

`EndIf`

D. `If` $d_p(v_i) = l$
`then` $l \leftarrow -1$ (no more paths will be obtained from p)
`else` $v_i \leftarrow$ following node in ${}^l p$
`EndIf`

`Until` (p_{sv_i} has a loop) or ($l = -1$)

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

`EndWhile`

Now some comments on conditions that affect the flow of instructions of KD-nd, relative to algorithm KD, will be made.

Obviously step 9c has only the additional constraint, $disj_V(p)$, that no node of p is used. Step 9(f)iiB of the algorithm assures that only potentially selectable disjoint paths (or paths that can possibly originate potentially selectable disjoint paths) with at most D arcs paths are added to X. The condition, "(head of a_{h+1} is marked *used*)", not present in algorithm KD, establishes that new paths will only be added to the set of candidate paths if all nodes up to (and

including) the head of its deviation arc are not used – with the exception of the originating node of the path; this guarantees that a path added to X is disjoint with previously selected paths or/and can possibly originate new paths which are disjoint with the ones already identified. Finally, variable l in KD [GMC01] gives the maximum depth of the nodes that can be deviation node of new paths – for the purpose of generating paths with at most D arcs; step 9e in KD-nd redefines this depth taking into consideration that a used node cannot be a deviation node.

5 Computational results

Two types of networks[1], were considered: low density networks (number of possible arcs is only 25% of the maximum number of arcs) and fully meshed networks for testing KD-nd. The former type arises in the context of transmission networks and the latter in the context of teletraffic networks. The normal stopping conditions were: P_c (attained end-to-end reliability) ≥ 0.995; or $K = 10$; or if $r_{st}(S_k) - r_{st}(S_{k-1}) < 1E-6$ and at least three paths had already been selected then p_k was the last path to be considered. However, if the first selected path has a reliability greater than P_d (desired end-to-end reliability) a second path is also selected in order to increase network survivability. Links reliability, $p_L(l)$, were randomly generated in two intervals $[0.9, 0.999]$ and $[0.99, 0.9999]$, marked as (a) and (b) in figure 3 and 1. Nodes reliability, $p_V(v)$ was randomly generated in the interval $[0.99, 0.99999]$.

For testing KD-ld three types of undirected networks were considered: fully meshed networks, high density networks (number of possible arcs is 75% of the maximum number of arcs) and medium density (number of possible arcs is 50% of the maximum number of arcs). These type of networks arise frequently in the context of teletraffic networks, and the symbol $d = 1, 0.75, 0.5$ will be used in the figures to identify the different density networks. The normal stopping conditions were: P_c (attained end-to-end reliability) $\geq Pd$; or $K = 20$; or $r_{st}(S_k) - r_{st}(S_{k-1}) < 1E-7$ and at least three paths have already been selected. However, if $P_c \geq Pd$ just after selecting the first path, a second path is also selected in order to increase network survivability. Link reliability, $p_L(l)$, was randomly generated in two intervals $[0.9, 0.99]$ and $[0.9, 0.999]$. The number of selected paths needed to satisfy the considered stopping conditions, oscillated between 2 and 4 (where the most effective stopping condition was $P_c \geq 0.99999$ for $p_L(l) \in [0.9, 0.99]$ the number of paths varied mostly between 3 and 4 and for $p_L(l) \in [0.9, 0.999]$ it varied mostly between 2 and 3).

5.1 Algorithm KD-ld

The proposed algorithm (as KD and MPS) spends most of its CPU effort in steps 4 and 6 because the number of selected states is very small (2 to 4) for the previously mentioned stopping conditions; therefore the CPU time presented in figures 1(a) and (b) is almost entirely due to those two steps.

[1]The used program for network generation was kindly borrowed by José Luis Santos.

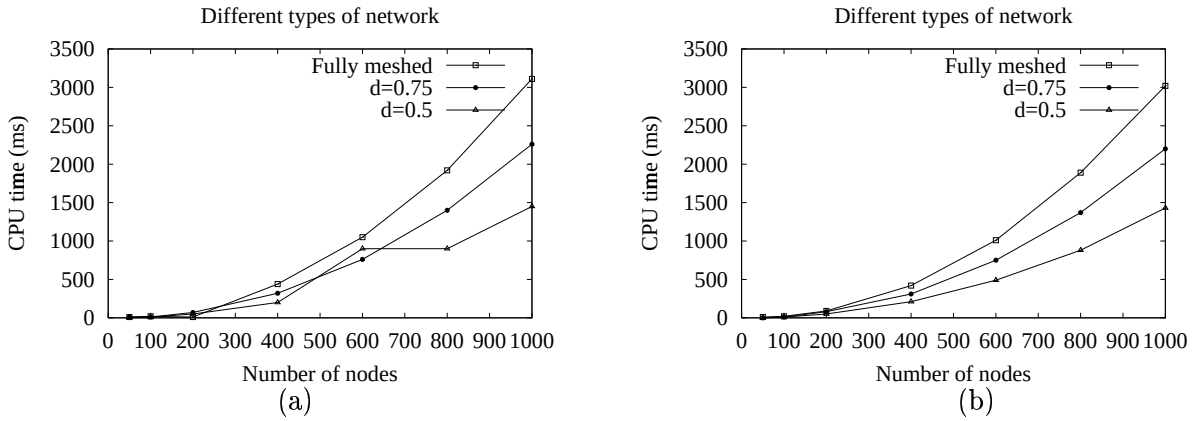

Figure 1: Algorithm KD-ld performance for normal stoping conditions (d is the network density): (a) for $D = 3$ and $p_L(l) \in [0.9, 0.999]$; (b) for $D = 5$ and $P_L(l) \in [0.9, 0.99]$

Experiences were made with $D = 3, 4, 5$, but they presented very similar CPU times (the curves would overlap in almost every point) regardless of which s-t pair was selected, under normal stopping conditions. Little influence was also detected due to the two considered intervals for $p_L(l)$. Therefore in figure 1(a) results are presented only for $D = 3$ and $p_L(l) \in [0.9, 0.999]$ and in figure 1(b) for $D = 5$ and $p_L(l) \in [0.9, 0.99]$.

The only features which clearly influence the performance of the algorithm are the graph density, d, and the number of nodes: more arcs leads to more computational effort in obtaining $\mathcal{T}_t^*$ and the sorted forward star form. KD-ld apparent indifference to D and $p_L(l)$ (in figures 1 (a) and (b)) resulted from the low number of selected paths. Note that the number of selected paths depends on the s-t selected pair.

In teletraffic networks it is sometimes desirable to have a set of selected disjoint paths larger than the ones obtained using the previously described stopping conditions. For the purpose of evaluating the performance of the algorithm in those situations, 20 disjoint paths (whenever they existed) were selected between node pairs. In figure 2 the corresponding CPU times are presented for $D = 3$, $D = 5$, $d = 1, 0.5, 0.8$, $p_L(l) \in [0.9, 0.99]$; in 2(a) CPU times are average values for various s-t pairs, and in 2(b) are the average CPU times for obtaining 20 disjoint paths for each node pair in the set of all s-t pairs, with t fixed (for several different end nodes t).

Some influence due to D can now be perceived in 2(a) and in 2(b); in the later case the CPU time presents some variability which is related do the simultaneous combination of allowing relatively long paths (5 arcs) in undirected networks with medium density. Figure 2(b) confirms that the proposed algorithm does indeed spend most of its CPU effort in steps 4 and 6. When the destination node was fixed and 20 paths were obtained for all s-t node pairs (with fixed t), the CPU times were greatly improved – the overhead is equally divided among all the considered node pairs, with the same destination node, t. These results confirm the efficiency of the proposed

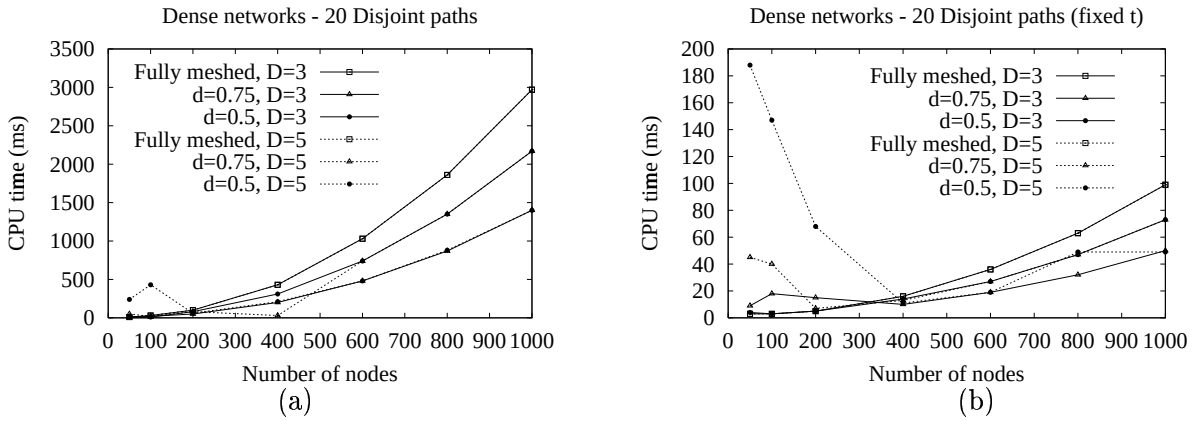

Figure 2: Obtaining the 20 most reliable disjoint paths for $p_L(l) \in [0.9, 0.99]$: (a) average for different s-t pairs; (b) average for all s-t pairs, with t fixed (for different t nodes).

approach.

The presented CPU times for KD-ld where obtained on a Bi-Pentium III at 866 MHz, with 512 MB of memory, under Linux.

5.2 Algorithm KD-nd

KD-nd also spends most of its CPU effort in steps 4 and 6, because the number of selected states is very small (2 to 6) for the previously mentioned stopping conditions; therefore the CPU time presented in figure 3(a) is almost entirely due to those two steps.

Experiences were made with $D = 3, 4, 5$, but they presented very similar CPU times (the curves would overlap in almost every point) regardless of which s-t pair was selected – so in figure 3(a) results are presented only for $D = 5$. Little influence was also detected due to the two considered intervals for $p_L(v)$, as can be confirmed in figure 3(a) and (b). The only features which clearly influence the performance of the algorithm are the graph density and the number of nodes: more arcs leads to more computational effort in obtaining $\mathcal{T}_t^*$ and the sorted forward star form. KD-nd apparent indifference to D and $p_V(l)$ resulted from the low number of selected paths.

The number of selected paths depends strongly on the s-t select pair and also on the interval for $p_V(l)$. A lower number of paths was usually required (for the tested s-t pairs) for networks with $p_V(l)$ in (2).

In figure 3(b) CPU times are presented when 50 paths (whenever they exist) are obtained for $D = 3$ and $D = 5$, to confirm the efficiency of the proposed approach, if a larger number of paths was desired.

The presented CPU times where obtained on a Pentium III at 500 MHz, with 256 MB of memory, under Linux.

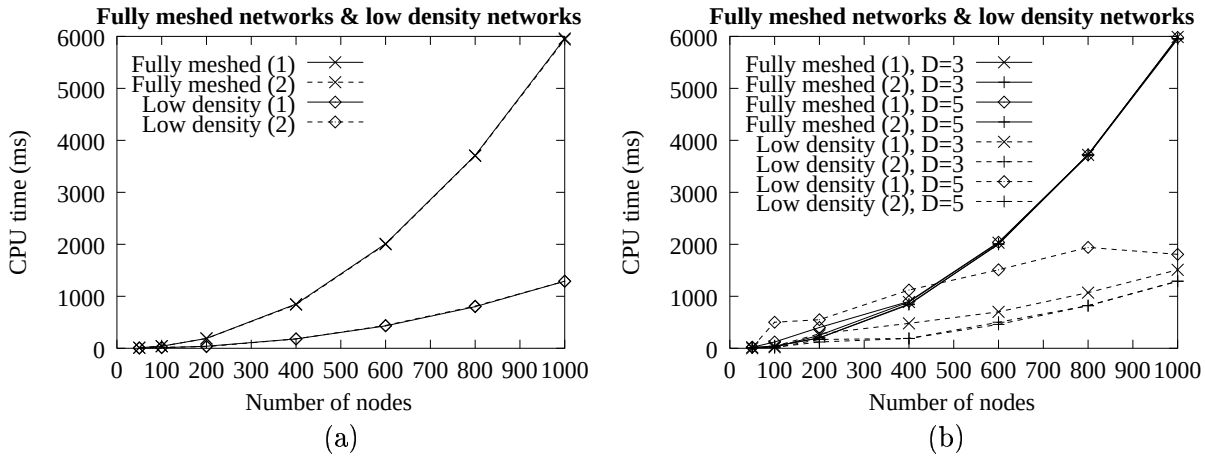

Figure 3: Results for KD-nd with (1) $p_V(l) \in [0.9, 0.999]$; (2) $P_V(l) \in [0.99, 0.9999]$ (a) $D = 5$, $2 \leq K \leq 10$, $P_d = 0.995$, $r_{st}(S_k) - r_{st}(S_{k-1}) < 1E-6$ (b) $K = 50$ and $D = 3, 5$

6 Conclusion

Very efficient algorithms which allows paths with a maximum number of arcs, to be obtained by decreasing probability of being operational until a given end-to-end reliability is achieved (whenever possible) has been presented.

This approach is different from others in the literature. Suurballes's algorithm, does not guarantee a maximum number of arcs in each selected path. On the other hand Torrieri's approach seeks to obtain optimal (in a certain sense) short disjoint paths, regardless of their reliability.

In algorithm KD-nd, with the stopping conditions described in section 5, it was experimentally verified that the number of obtained paths oscillated between 2 and 6; in algorithm KD-ld, also with the stopping conditions described in section 5, it was experimentally verified that the number of obtained paths oscillated between 2 and 4. This is not irrelevant from the point of view of a network operator wishing to guarantee a certain grade of service, at the lowest cost. Network reliability and survivability is improved if adequate sets of node disjoint paths are used between source and terminal nodes in the network. Selecting disjoint paths by decreasing order of their reliability will guarantee that we are using an adequate set of disjoint paths from a reliability point of view.

The initial cost of steps 4 and 6 of KD-ld and KD-nd, is a sort of overhead of the algorithms, but it can be minimised if the built in structures are re-utilised to obtain the disjoint paths from all nodes to t, with very little additional computational effort.

In the context of teletraffic networks it may be more interesting to obtain link disjoint paths because node failure probability is usually rather low, in these networks. An algorithm for this purpose, KD-ld was proposed.

References

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms and applications*. Prentice Hall, 1993.

[Bal79] M. O. Ball. Computing network reliability. *Operations Research*, 27(4):821–838, 1979.

[DGKK79] R. Dial, F. Glover, D. Karney, and D. Klingman. A computational analysis of alternative algorithms and labeling techniques for finding shortest path trees. *Networks*, 9:215–348, 1979.

[DML94] R. D. Doverspike, J. A. Morgan, and W. Leland. Network design sensitivity studies for use of digital cross-connect systems in survivable network architectures. *IEEE Journal on Selected Areas in Communications*, 12(1):69–78, January 1994.

[Epp98] D. Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28:652–673, 1998.

[GMC01] T. Gomes, L. Martins, and J. Craveirinha. An algorithm for calculating the k shortest paths with a maximum number of arcs. Accepted for publication in the Journal *Investigação Operacional*, 2001.

[KDP95] K. R. Krishnan, R. D. Doverspike, and C. D. Pack. Improved survivability with multi-layer dynamic routing. *IEEE Communications Magazine*, 33(33):62–68, July 1995.

[MPS99a] 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.

[MPS99b] 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.

[San96] I. Saniee. Optimal routing designs in self-healing communications networks. *International Transactions on Operations Research*, 3(2):187–195, 1996.

[Suu74] J. W. Suurballe. Disjoint paths in networks. *Networks*, 4:125–145, 1974.

[Tor92] D. Torrieri. Algorithm for finding an optimal set of short disjoint paths in a communication network. *IEEE Transactions on Communications*, 40(11):1698–1702, November 1992.

[Tor94] D. Torrieri. Calculation of node-pair reliability in large networks with unreliable nodes. *IEEE Transactions on reliability*, 43(3):375–377, September 1994.