Paper

A new algorithm for calculating the most reliable pair of disjoint paths in a network

Teresa Gomes, José Craveirinha, and Artur Violante

Abstract— In various types telecommunication networks, namely mobile ad hoc networks, WDM networks and MPLS networks, there is the necessity of calculating disjoint paths for given node to node connections in order to increase the reliability of the services supported by these networks. This leads to the problem of calculating a pair of disjoint paths (or a set of disjoint paths) which optimises some measure of performance in those networks. In this paper we present an algorithm, designated as OptDP, for obtaining the most reliable pair of disjoint paths based on the loopless version of MPS, a very efficient *k*-shortest path algorithm, and on Dijkstra algorithm. Since to the best of our knowledge there is no other proposal of an algorithm capable of solving exactly the same problem we perform a comparison with the application to this problem of the DPSP algorithm which calculates a set of disjoint paths with high reliability. Also a comparison with a simplified version (designated as NopDP) of the proposed algorithm, which stops after a maximal number *F* of candidate pairs of paths have been found, is presented. The comparison also includes the percentage of cases in which both algorithms were not capable of finding the optimal pair.

Keywords— reliability, OR in telecommunications, routing.

1. Introduction

In various types telecommunication networks, namely mobile ad hoc networks, wavelength division multiplexing (WDM) networks and multiprotocol label switching (MPLS) networks, there is the necessity of calculating disjoint paths for given node to node connections in order to increase the reliability (hence improving the quality of service – QoS) of the services supported by these networks. This leads to the problem of calculating a pair of disjoint paths (or a set of disjoint paths) which optimises some measure of performance in those networks.

Performance measures are generally defined having as basis the cost of the arcs. If the metric associated with each arc and the function to be optimised are additive measures, the calculation of a set of disjoint paths can be made using the well known algorithm proposed in [10]. If only a pair of paths is required, then the efficient polynomial time algorithm proposed by Suurballe and Tarjan [11] will solve the problem. If the most reliable pair of paths is desired, the algorithm in [11] cannot be used because the reliability of the union of the two paths is not additive in the path reliabilities.

The problem of finding the set of paths which maximises the two-terminal (ie., node-to-node) reliability metric has not received much attention according to Papadimitratos *et al.* [9]. These authors propose an algorithm (with polynomial worst case complexity) which calculates a set of disjoint paths (without length constraints) with high reliability, that was applied in the context of mobile ad hoc networks (MANET).

In [5] an algorithm is proposed for obtaining k disjoint paths between two different nodes, s and t, in a network with k different costs on every edge, such that the total cost of the paths is minimised (where the jth edge-cost is associated with the jth path).

The approach used in the present work has similarities with the enhancement of the two-step-approach [4] and with the iterative two-step-approach (ITSA) algorithm [8] for optimal diverse routing with shared protection in connection-oriented networks, where the arc costs of the protection path depend on the selected working path. Note that, in our case, the cost of the protection path does not depend on the selected working path, and the function to be optimised is not a linear combination of the costs of the working and protection paths, unlike the problem in [8].

In this paper we present an algorithm, designated as OptDP, for obtaining the most reliable pair of disjoint paths based on the loopless version of the k-shortest paths algorithm MPS [7] and on Dijkstra algorithm. The basic structure is analogous to the one of the algorithm RLDPC-BF proposed by the authors in [2] for calculating the most reliable pair of disjoint paths with a maximum number of arcs per path, based on KD (k-shortest paths with at most D arcs [3]) and Bellman-Ford algorithms.

Since to the best of our knowledge there is no other proposal of an algorithm capable of solving exactly the same problem we performed a comparison with the disjoint path selection protocol (DPSP) algorithm [9] which calculates a set of disjoint paths with high reliability. Note that this algorithm does not guarantee the determination of the optimal set of disjoint paths in terms of reliability. We compared the proposed algorithm with DPSP when this is used to obtain a pair of disjoint paths with high reliability DPSP(k = 2), in terms of central processing unit (CPU) time, for four sets of test networks, with low and high reliability in the edges, a number of nodes (n) varying from 50 to 500 and number of edges, $m = 3n$ and $m = 2n$. Also a comparison with a simplified version (designated as NopDP) of our algorithm which stops after a maximal number F of candidate

pairs of paths has been found, is presented (the used implementation considers $F = 5$). The comparison also includes the percentage of cases in which both algorithms were not capable of finding the optimal pair.

The major conclusion of the computational experiments with the test networks was that the simplified version of the proposed algorithm (NopDP) always performed better than DPSP(k = 2) in terms of CPU and enabled a significant larger relative number of optimal pairs of paths to be obtained. As for the exact algorithm (OptDP) it is less efficient than DPSP(k = 2) for networks with high reliability but becomes more efficient for networks with low reliability (this improvement is particularly significant in networks with $m = 3n$).

The paper is organised as follows. In Section 2 the problem will be formalised and in Section 3 the proposed algorithm is described. In Section 4 the DPSP algorithm is shortly reviewed. Experimental results from the three algorithms are shown and discussed in Section 5, followed by some final remarks in Section 6.

2. Problem formulation

Let $G = (N,L)$ be a directed graph where $N = \{v_1, v_2, \ldots, v_n\}$ 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. Let $l = (i, j)$ be an arc where 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 loopless condition is implicitly assumed. The word "loopless" will continue to be omitted until explicit reference is needed.

Each link $l \in L$ has a probability $p_L(l)$ of being operational. Nodes are assumed not to fail. In a network where links fail (independently) and one seeks link disjoint paths from s to t, 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 in a path:

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

The cost of a path $p = \langle s,(s,v_1),v_1,\ldots,(v_w,t),t\rangle$ is $\mathcal{C}(p) = \sum_{(v_i,v_j)\in p} c_{v_iv_j}$, and its reliability is:

$$\Pr(p) = e^{-\mathcal{C}(p)}, \qquad (2)$$

where $\Pr(p)$ represents the probability of path p being operational. Equation 2 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 is equivalent to enumerating, by decreasing order of their reliability, the k most reliable paths.

The most reliable pair of link disjoint paths (p_w, p_v) has a reliability given by:

$$\max_{p_w,p_v} \Pr(p_w \cup p_v) = \Pr(p_w) + (1 - \Pr(p_w))\Pr(p_v), \qquad (3)$$

where p_w and p_v are the working and protection paths, respectively. As can be seen from Eq. (3) $\Pr(p_w \cup p_v)$ cannot be written as a linear function of the costs of p_w and p_v. Two disjoint paths may have minimum $C(p_w) + C(p_v)$ but they may not be the paths with maximal $\Pr(p_w \cup p_v)$.

3. Description of OptDP

The sequential generation of paths p_i (selected by decreasing reliability order) can be made by using any k-shortest path ranking algorithm. In this work the loopless version of MPS algorithm was chosen [7], due to its efficiency [6].

For each i-shortest path p_i (where i represents the order of a selected path – p_i is a candidate working path) there may exist more than one link disjoint path (p_j, a candidate protection path for p_i). The path p_j which maximises $\Pr(p_i \cup p_j)$ (with p_i fixed) will be the one with highest reliability among all the feasible paths; therefore a subalgorithm is needed for efficiently obtaining the most reliable path disjoint with p_i. This algorithm can simply be the Dijkstra algorithm applied to graph G with the links in p_i (temporarily) removed, the algorithm execution being stopped as soon as the destination note t is selected as a minimum distance node.

The proposed algorithm, designated by OptDP, requires a condition to detect that the calculated disjoint path pair is optimal.

Suppose that for each path p_w, the most reliable link disjoint path p_v was obtained, such that at any given step of the algorithm the only recorded pair of paths is the one with the highest $\Pr(p_w \cup p_v)$. Considering that the next (most reliable) path, generated by the k-shortest path subalgorithm, to be selected in the main algorithm is p_i ($i > w$) such that:

$$\Pr(p_i) + (1 - \Pr(p_i))\Pr(p_i) \leq \Pr(p_w) + (1 - \Pr(p_w))\Pr(p_v), \qquad (4)$$

then (p_w, p_v) is the pair of paths with maximal reliability. The verification of this statement is straightforward. Let p_j be the most reliable path link disjoint with p_i, if $\Pr(p_j) \leq \Pr(p_i)$ then any other pair of paths obtained from this point onwards will always have reliability less than $\Pr(p_i) + (1 - \Pr(p_i))\Pr(p_i)$ therefore lower than $\Pr(p_w \cup p_v)$; if $\Pr(p_j) > \Pr(p_i)$ then p_j was previously generated and the reliability of the corresponding pair was not greater than the one of the current best pair, thence this case is irrelevant. Note that this optimal stopping condition (4) is the same as in the algorithm [2] proposed by the authors for calculating the most reliable pair of disjoint paths with length constraints.

Having established the optimal stopping rule of the algorithm (OptDP), its flowchart is presented in Fig. 1. The ex-

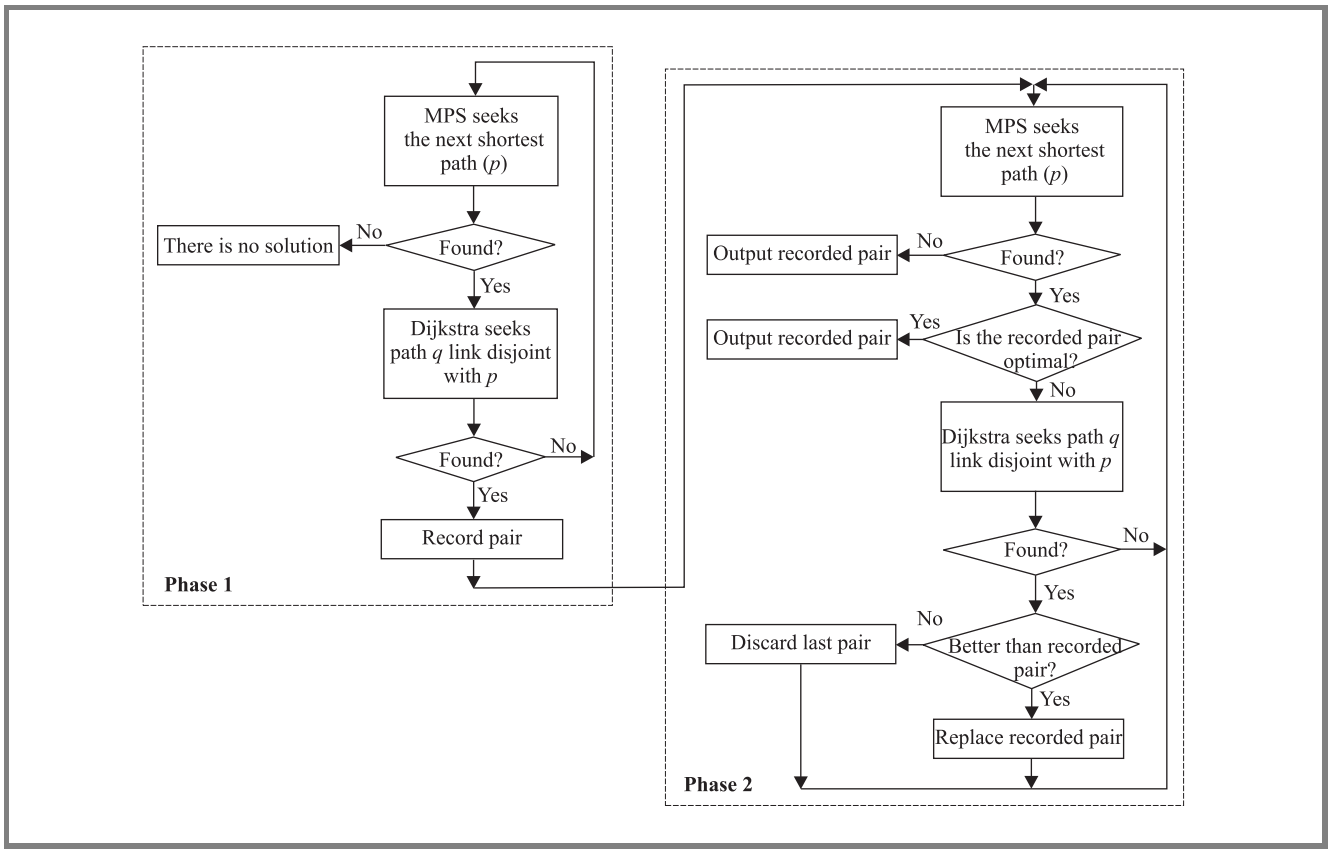

Fig. 1. Flowchart of OptDP.

perimental results will show that, although the optimal disjoint path is frequently among the first ones which are obtained by OptDP, sometimes it is difficult to verify the optimal stopping condition. So a variant of the algorithm was implemented, NopDP, which counts the number of generated path pairs and stops when the optimal stopping condition is satisfied or when the number of generated path pair reaches a pre-established value F, returning the current best pair of disjoint paths.

The main structure of the algorithm has two phases shown in Fig. 1: the obtainment of the first pair of link disjoint paths and the search and/or detection of the optimal pair of paths. In the first phase the algorithm may terminate without finding a solution: no single link disjoint pair of paths was identified (in a connected network this situation can only occur if the network is 1-line-connected).

Having completed phase 1 (we assume that at least a disjoint path pair is always found) and having recorded a pair of link disjoint paths, the second phase of the algorithm consists of improving this solution (whenever possible) until either the recorded pair of paths is detected to be optimal, or no more (working) paths can be found. This last condition implies that the best recorded pair is in fact the optimal one.

If the graph $G^u(N,l)$ which represents the structure of a telecommunications network is undirected, the proposed algorithm can still be used. Each (undirected) edge is replaced by two directed arcs in opposite directions with cost equal to the cost of the edge and the corresponding directed version, G, of G^u is used by OptDP. All the edges of a working path p_w have to be removed (temporarily) from the network graph before running the Dijkstra algorithm; in this case this is done by removing (temporarily) from G all arcs in p_w and also the corresponding arcs in opposite direction.

4. A brief overview of DPSP algorithm

The DPSP algorithm, by Papadimitratos *et al.* [9], iteratively builds a set of disjoint paths of high reliability, for undirected networks. The implementation of DPSP uses the directed graph corresponding to the undirected network under analysis.

Let us assume that at given step of the algorithm, k disjoint paths have already been obtained and are stored in set D_k ($D_k = \cup_{i=1}^{k}\{p_i\}$). The arcs which belong to the paths in set D_k are (temporarily) removed form the network graph, making their cost equal to ∞. The reverse arcs, that is the arcs in opposite direction corresponding to the arcs in D_k (recall that each edge in G^u is represented in G by two arcs in opposite directions) have its cost (temporarily) set to its symmetrical value. Using a shortest path algorithm which works with negative arc costs (in a graph without negative cycles) the more reliable path, p_c, is found in this modified graph and it is the candidate path which

Table 1
Test networks

n		50	100	150	200	250	300	350	400	450	500
$m=2n$	$d(G)$	5-7	7-8	8-9	8-10	8-10	9-10	9-11	9-11	10-11	10-12
	$\bar{d}(u,v)$	4.0	4.6	5.0	5.2	5.4	5.5	5.6	5.7	5.9	5.9
$m=3n$	$d(G)$	4-5	5-6	6-7	6-7	6-7	6-7	7-8	7-8	7-8	7-8
	$\bar{d}(u,v)$	3.4	3.9	4.1	4.3	4.4	4.5	4.6	4.7	4.7	4.8

Explanations: n – the number of nodes, m – the number of arcs, $d(G)$ – the network diameter, $\bar{d}(u,v)$ – the average node distance.

enables D_{k+1} to be obtained from D_k. If no interlacing exists, that is if the candidate path has no arcs with negative costs, which means p_c is disjoint with all the paths in D_k, then p_c is added to the set: $D_{k+1} = D_k \cup \{p_c\}$. If an interlacing exists, the algorithm will evaluate whether the removal of this interlacing and the corresponding change in the set of disjoint paths will lead to an increase in its reliability. Let A be the sub-set of paths in D_k which interlace with the candidate path p_c. Let I be the interlacing between A and p_c (the sub-set of arcs in paths in A the reverse arcs of which belong to p_c). Let B be the set of paths which would result from the removal of that interlacing (for details see [9]), then (by omission the algorithm does not remove the interlacing I):

1) $P_{op} = \Pr(p_c) \times \prod_{l \in I} p_L(l)$,
2) $m_1 = 1 - (1 - P_{op}) \prod_{p_i \in A} (1 - \Pr(p_i))$,
3) $m_2 = 1 - \prod_{p_j \in B} (1 - \Pr(p_j))$,
4) if $m_1 < m_2$ then remove the interlacing I.

According to [9] m_1 captures the reliability of the original path set and the candidate shortest path (p_c) and m_2 the the reliability of the original path set after removing the interlacing. If m_1 is greater than (or equal to) m_2 the interlacing is not removed. In this case, one or more arcs in I are removed from the graph (their costs is set to ∞) and their original costs are stored in a list. This will ensure that the next candidate path will be different from p_c. If $m_1 < m_2$ then the interlacing I is removed and the costs of the arcs in the interlacing recover their original values; the corresponding reverse arcs also recover their original costs. This procedure is repeated until no more candidate paths exist.

In the implemented version of DPSP the modified version of Dijkstra, as described in [1], was used for obtaining the shortest path between a pair of nodes in a network with negative costs. When the interlacing is not removed, all the arcs in the interlacing are removed from the graph.

The DPSP algorithm can be used for obtaining a pair of disjoint paths of high reliability, stopping when D_2 is obtained. This version of DPSP will be designated by DPSP(k = 2). The DPSP(k = 2) algorithm starts by obtaining the most reliable path which is stored in set D_1. If a candidate path p_c is obtained such that no interlacing exists between D_1 and p_c then $D_2 = D_1 \cup \{p_c\}$; if an interlacing I is obtained between D_1 and p_c, the metrics m_1 and m_2 are calculated and if the interlacing is removed D_2 is obtained, otherwise the algorithm proceeds by changing arc costs so that a new candidate path p_c will possibly be obtained.

5. Experimental results

Results are presented for undirected networks, with low connectivity, as indicated in Table 1. These types of features are common in wavelength division multiplexing (WDM) optical networks. For each number of nodes n, ten different networks were randomly generated[1] with the same number of arcs and nodes; the arc reliabilities were randomly generated in $[1-5\cdot10^{-4}, 1-10^{-6}]$ and $[0.8, 0.99]$. The first range of reliability values is adequate for WDM networks and the second for mobile ad hoc networks. Two different network densities were used: $m = 2n$ and $m = 3n$. In order to capture as faithfully as possible the algorithm dependences on the range of link reliabilities, the networks were obtained as follows. Firstly two sets of networks, for $m = 3n$ and $m = 2n$, and $p_L(l) \in [1-5\cdot10^{-4}, 1-10^{-6}]$, were obtained (as already mentioned, for each value of n, 10 networks were randomly generated). Secondly, using the same topological structure of the previous 2×100 networks, two new sets were obtained where the link costs were randomly generated in the range $[0.8, 0.99]$.

Due to the low network connectivity a great variation in CPU time used by OptDP was observed depending on the *s-t* pair. Therefore for each network a pair of disjoint paths was seeked for all $(n \times (n-1))$ node pairs[2] and the average CPU time obtained per pair of disjoint paths for each node pair in the set of all *s-t* pairs with t fixed (for all nodes t). This allows MPS (and therefore OptDP) to re-use the tree of shortest paths from all nodes to t and the ordered set of the network arcs.

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

[2] Due to the nature of MPS the cost of obtaining the optimal disjoint pair from s to t and from t to s is not identical.

Finding a pair of disjoint paths was easy, but detecting the optimality condition was sometimes difficult (in the sense that a large number of paths had to generated) but it was always successfully achieved by OptDP. So although OptDP has to generate a significant number of pairs of disjoint paths, in order to verify the optimality stopping condition, it was verified, in the test networks, that the optimal path was one of the first four paths (for networks with $m = 3n$) in 99% of the cases on average (97% was the lowest value obtained for all test networks). In Fig. 2 it

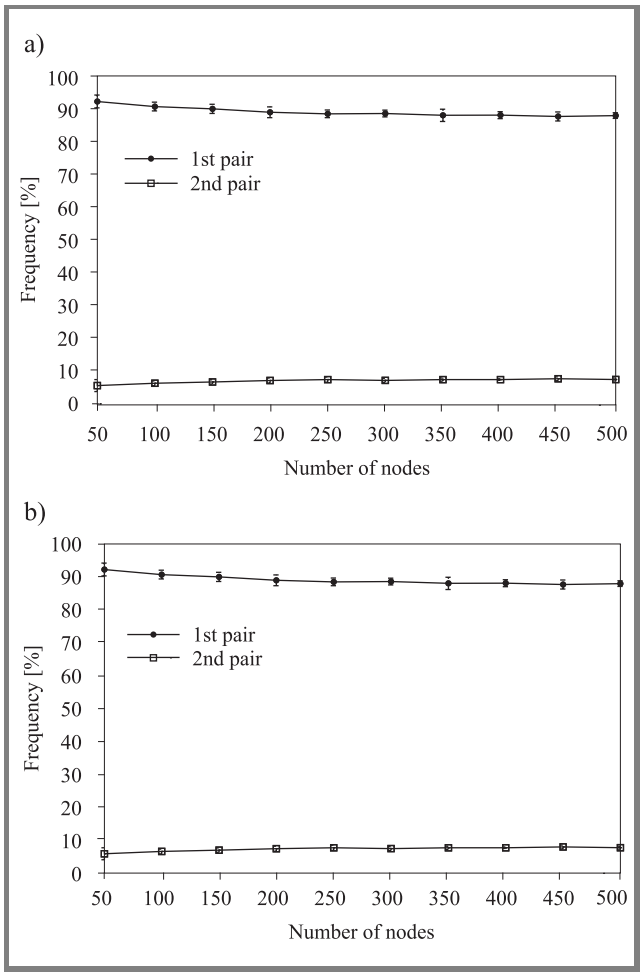

Fig. 2. Frequency of cases, where the optimal pair is either the first or second pair for $m = 3n$. (a) $p_L(l) \in [1-5\cdot10^{-4}, 1-10^{-6}]$; (b) $p_L(l) \in [0.8, 0.99]$.

can be seen that the first and second pairs are the optimal paths most of the times (results for networks with $m = 2n$ are similar and therefore are not presented). On average the third and fourth pairs are the optimal ones for at most 5% of the cases. For all tested cases the first pair, calculated by OptDP, is optimal with a frequency between 85% (for $m = 3n$ and high reliability networks) and 96% (for $m = 2n$ and high reliability networks) of the cases and the second pair is the optimal one with a frequency between 3% and 12% (both upper and lower bounds were obtained in networks with high reliability, with $m = 2n$).

Based on these results, which strongly suggest that the first four pairs represent a great percentage of the total number of optimal pairs, it was decided to implement a "shorter" version of the algorithm, NopDP. This algorithm is similar to OptDP but will sometimes return path pairs the optimality of which was not confirmed. Results will be presented for NopDP, when $F = 5$, which means the algorithm either stops because the 5th pair was obtained, or because it was not necessary to generate more than 4 path pairs before detecting that the optimality condition was true.

The CPU times per node pair are presented in Figs. 3 and 4 – the PC used was a Pentium IV at 2.8 GHz and 500 Mb of RAM. The average values (per network) obtained by OptDP presented some variation and therefore an error bar was added, centred in the average μ of the collected samples (one sample per network) which goes from

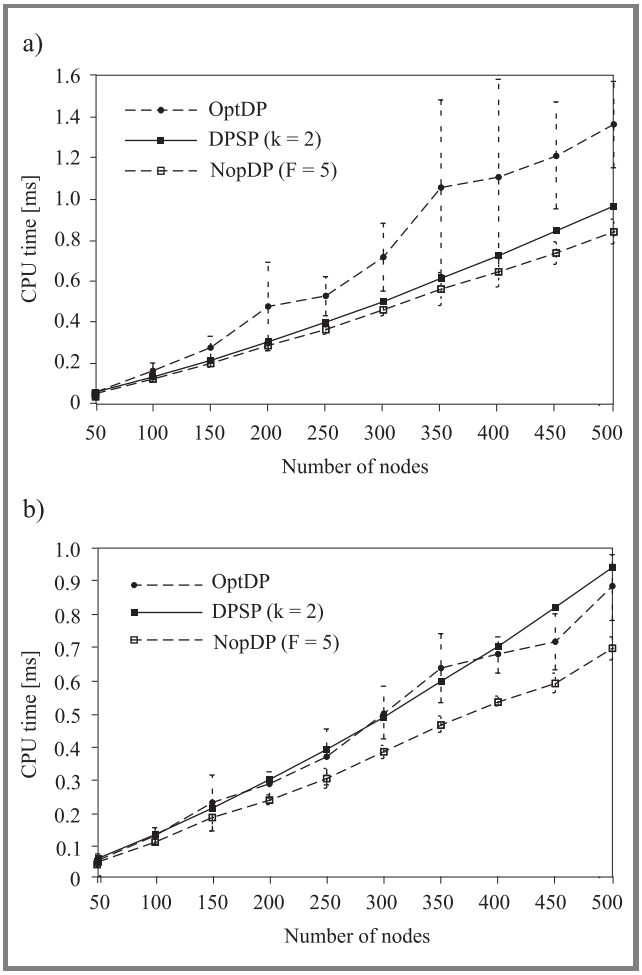

Fig. 3. CPU time per pair of nodes in the networks for $m = 3n$. (a) $p_L(l) \in [1-5\cdot10^{-4}, 1-10^{-6}]$; (b) $p_L(l) \in [0.8, 0.99]$.

$\max(0, \mu-\sigma)$ to $\mu+\sigma$, where σ is the standard deviation of the sample. The purpose of this bar was to show the variability of the results in the case of OptDP. The DPSP(k = 2) algorithm does not present significant variation, therefore no error bar was added in this case. An error bar was

also added to the results of NopDP(F = 5) to show that the variability in this case is rather small, when compared to OptDP.

In Figure 3 CPU times for networks with $m = 3n$ for low and high edge reliability, are presented. Figure 3a shows that OptDP is the less efficient algorithm for $p_L(l) \in [1-5\cdot10^{-4}, 1-10^{-6}]$. However, from Fig. 3b with $p_L(l) \in [1-5\cdot10^{-4}, 1-10^{-6}]$ it is not clear which algorithm is the less efficient, because the line for OptDP is interlaced with the line for DPSP(k = 2). On the other hand NopDP(F = 5) is consistently more efficient than OptDP and DPSP(k = 2).

In the case of networks with $m = 2n$ (see Fig. 4) OptDP continues to be the less efficient approach for $p_L(l) \in [1-5\cdot10^{-4}, 1-10^{-6}]$ and although it improves its relative performance for $p_L(l) \in [0.8, 0.99]$, it remains the less efficient approach (except for smaller networks: $n = 50, 100, 150$). On the other hand NopDP(F = 5) continues to be the best approach, as far has CPU per node pair is concerned.

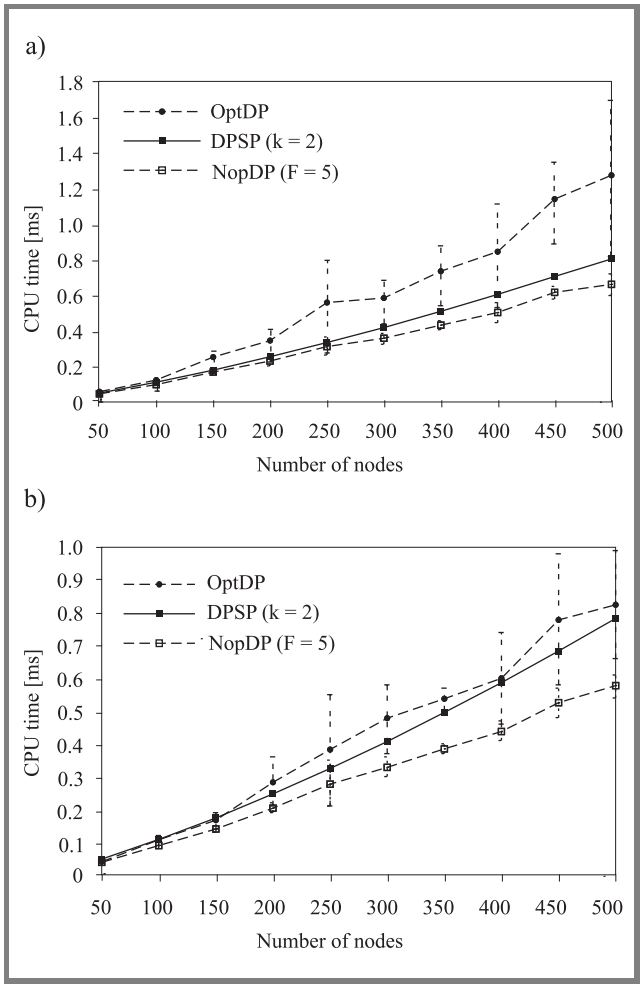

Fig. 4. CPU time per pair of nodes in the networks for $m = 2n$. (a) $p_L(l) \in [1-5\cdot10^{-4}, 1-10^{-6}]$; (b) $p_L(l) \in [0.8, 0.99]$.

It should be noted that when DPSP(k = 2) (or DPSP) is used, the algorithm has no way of knowing whether the optimal pair (set) of disjoint paths has been obtained. NopDP on the other hand knows that some of its solutions are indeed optimal (the remaining might be optimal, but NopDP did not run long enough to confirm their optimality or they may be sub-optimal). In Table 2, the average frequency of NopDP(F = 5) termination because an optimal solution was found, is presented. The minimal and maximal values obtained in all experiments were 68.3% and 98.9%.

Table 2

NopDP(F = 5) exits with the detection of the optimal condition

n	$m = 3n$		$m = 2n$	
	A	B	A	B
50	93.8	95.5	91.6	95.0
100	88.8	92.6	89.3	91.0
150	87.3	91.2	85.0	90.8
200	84.8	91.5	85.8	88.6
250	85.1	91.5	83.2	88.5
300	83.6	90.1	84.2	86.9
350	81.6	88.8	83.6	87.1
400	83.3	89.8	83.7	88.8
450	82.8	91.2	81.2	86.3
500	83.3	89.3	82.7	87.6

Explanations: A – high reliability, $p_L(l) \in [1-5\cdot10^{-4}, 1-10^{-6}]$; B – low reliability, $p_L(l) \in [0.8, 0.99]$.

Finally to confirm which of the algorithms, DPSP(k = 2) or NopDP(F = 5) did obtain the greater number of optimal solutions, an analysis was made (based on the optimal reliability value obtained by using OptDP) of the values returned by DPSP(k = 2) and NopDP(F = 5) (using 12 significant digits). Observing the results in Table 3 the first observation is that DPSP obtains sub-optimal solutions in 1.1%–2.9% of the cases while NopDP only fails in less than 1% of the cases for high reliability networks and in less than 0.5% of the cases for low reliability networks. The average values of the relative differences of the reliability of the obtained sub-optimal solution with respect to the optimal one are shown in Tables 3 and 4. These differences are in the same range for both algorithms, with a slight increase for NopDP(F = 5) in the case of less reliable networks.

Results for networks with $m = 2n$, regarding the frequency of sub-optimal solutions, are presented in Table 4. For DPSP(k = 2) this frequency is in the range 2.0%–3.2% while NopDP(F = 5) continues to present values under 1% for high reliability networks and under 0.5% for low reliability networks (with one exception: 0.51%). The relative errors of reliability (for sub-optimal node pairs) for $m = 2n$ are greater than for $m = 3n$, for both DPSP(k = 2) and NopDP(F = 5). The relation between this relative error is around 3 when comparing NopDP(F = 5) and DPSP(k = 2) and the relative frequency of sub-optimal solutions is around 9 for low reliability networks and around 4 for

Table 3

Frequency of non optimal pairs obtained with DPSP(k = 2) and NopDP(F = 5), and corresponding average reliability relative error (of non optimal pairs) for $m = 3n$

	$m = 3n$ $p_L(l) \in [1-5\cdot10^{-4}, 1-10^{-6}]$					$m = 3n$ $p_L(l) \in [0.8, 0.99]$			
	DPSP(k = 2)		NopDP(F = 5)			DPSP(k = 2)		NopDP(F = 5)	
n	[%]	ΔPr	[%]	ΔPr	n	[%]	ΔPr	[%]	ΔPr
50	1.91	$1.6\cdot10^{-8}$	0.31	$4.9\cdot10^{-8}$	50	1.13	$1.9\cdot10^{-3}$	0.05	$7.0\cdot10^{-3}$
100	2.34	$1.6\cdot10^{-8}$	0.35	$4.7\cdot10^{-8}$	100	1.84	$2.2\cdot10^{-3}$	0.18	$5.8\cdot10^{-3}$
150	2.52	$1.6\cdot10^{-8}$	0.42	$4.7\cdot10^{-8}$	150	2.09	$2.3\cdot10^{-3}$	0.24	$5.9\cdot10^{-3}$
200	2.74	$1.7\cdot10^{-8}$	0.68	$5.1\cdot10^{-8}$	200	2.10	$2.3\cdot10^{-3}$	0.27	$6.1\cdot10^{-3}$
250	2.78	$1.6\cdot10^{-8}$	0.61	$4.7\cdot10^{-8}$	250	2.12	$2.4\cdot10^{-3}$	0.25	$6.7\cdot10^{-3}$
300	2.75	$1.7\cdot10^{-8}$	0.82	$5.0\cdot10^{-8}$	300	2.17	$2.5\cdot10^{-3}$	0.32	$7.1\cdot10^{-3}$
350	2.86	$1.7\cdot10^{-8}$	0.94	$4.6\cdot10^{-8}$	350	2.24	$2.5\cdot10^{-3}$	0.41	$7.1\cdot10^{-3}$
400	2.73	$1.7\cdot10^{-8}$	0.88	$4.9\cdot10^{-8}$	400	2.23	$2.7\cdot10^{-3}$	0.35	$7.3\cdot10^{-3}$
450	2.93	$1.8\cdot10^{-8}$	0.87	$5.2\cdot10^{-8}$	450	2.03	$2.6\cdot10^{-3}$	0.29	$7.0\cdot10^{-3}$
500	2.69	$1.8\cdot10^{-8}$	0.85	$5.6\cdot10^{-8}$	500	2.21	$2.6\cdot10^{-3}$	0.39	$7.0\cdot10^{-3}$

Table 4

Frequency of non optimal pairs obtained with DPSP(k = 2) and NopDP(F = 5), and corresponding average reliability relative error (of non optimal pairs) for $m = 2n$

	$m = 2n$ $p_L(l) \in [1-5\cdot10^{-4}, 1-10^{-6}]$					$m = 2n$ $p_L(l) \in [0.8, 0.99]$			
	DPSP(k = 2)		NopDP(F = 5)			DPSP(k = 2)		NopDP(F = 5)	
n	[%]	ΔPr	[%]	ΔPr	n	[%]	ΔPr	[%]	ΔPr
50	2.38	$4.9\cdot10^{-8}$	0.29	$8.7\cdot10^{-8}$	50	1.99	$4.4\cdot10^{-3}$	0.10	$9.8\cdot10^{-3}$
100	2.48	$4.3\cdot10^{-8}$	0.40	$1.35\cdot10^{-7}$	100	2.43	$5.9\cdot10^{-3}$	0.15	$1.47\cdot10^{-2}$
150	2.75	$4.6\cdot10^{-8}$	0.62	$1.22\cdot10^{-7}$	150	2.50	$5.1\cdot10^{-3}$	0.23	$1.31\cdot10^{-2}$
200	2.78	$4.7\cdot10^{-8}$	0.60	$1.33\cdot10^{-7}$	200	2.70	$5.3\cdot10^{-3}$	0.37	$1.38\cdot10^{-2}$
250	3.00	$5.1\cdot10^{-8}$	0.77	$1.46\cdot10^{-7}$	250	2.82	$5.4\cdot10^{-3}$	0.39	$1.24\cdot10^{-2}$
300	2.77	$5.0\cdot10^{-8}$	0.73	$1.47\cdot10^{-7}$	300	2.90	$6.0\cdot10^{-3}$	0.46	$1.38\cdot10^{-2}$
350	2.95	$5.1\cdot10^{-8}$	0.76	$1.56\cdot10^{-7}$	350	2.87	$5.9\cdot10^{-3}$	0.46	$1.45\cdot10^{-2}$
400	2.83	$4.8\cdot10^{-8}$	0.80	$1.44\cdot10^{-7}$	400	2.72	$6.0\cdot10^{-3}$	0.38	$1.36\cdot10^{-2}$
450	3.17	$5.3\cdot10^{-8}$	0.93	$1.48\cdot10^{-7}$	450	2.88	$5.9\cdot10^{-3}$	0.51	$1.41\cdot10^{-2}$
500	2.90	$5.3\cdot10^{-8}$	0.85	$1.56\cdot10^{-7}$	500	2.83	$6.1\cdot10^{-3}$	0.43	$1.46\cdot10^{-2}$

high reliability networks when comparing DPSP(k = 2)) and NopDP(F = 5). Therefore the results for NopDP(F = 5) are significantly more favourable than for DPSP(k = 2).

6. Conclusions

A new algorithm, OptDP, for obtaining the most reliable pair of edge disjoint paths, and a "shorter" variant, NopDP, which does not always guarantee the generated path pair is optimal, have been proposed. Algorithm DPSP was also reviewed and a "truncated" version DPSP(k = 2) was used for obtaining a pair of disjoint paths with high reliability.

The performances of OptDP, NopDP(F = 5) and DPSP(k = 2) were evaluated through numerous experiments for randomly generated networks, with different connectivities. For each value of connectivity two sets of networks were generated, one with low reliability and the other with high reliability.

These experiments enabled the good performance of OptDP to be put in evidence for less reliable networks when compared with DPSP(k = 2). In particular NopDP(F = 5) was shown to be a good compromise between precision (number of optimal solutions obtained) and required CPU time.

Acknowledgement

Work partially supported by programme POSI of the III E programme cosponsored by FEDER and national funds.

References

[1] R. Bhandari, *Survivable Networks, Algorithms for Diverse Routing*. Boston [etc.]: Kluwer, 1999.

[2] T. Gomes and J. Craveirinha, "Efficient calculation of the most reliable pair of link disjoint paths in telecommunication networks", in *Proc. Conf. DSTIS 2004*, Warsaw, Poland, 2004 (accepted for publication in *Eur. J. Oper. Res.*).

[3] T. Gomes, L. Martins, and J. Craveirinha, "An algorithm for calculating the *k*-shortest paths with a maximum number of arcs", *Invest. Oper.*, vol. 21, no. 2, pp. 235–244, 2001.

[4] P. Laborczi, J. Tapolcai, P.-H. Ho, T. Cinkler, A. Recski, and H. T. Mouftah, "Algorithms for assymetrically wheigted pair of disjoint paths in survivable networks", in *Proc. Des. Rel. Commun. Netw. DCRN 2001*, Budapest, Hungary, 2001, pp. 220–227.

[5] C.-L. Li, S. T. McCormick, and D. Simchi-Levi, "Finding disjoint paths with different path costs: complexity and algorithms", *Networks*, vol. 22, pp. 653–667, 1992.

[6] E. Martins, M. Pascoal, and J. Santos, "An algorithm for ranking loopless paths", Tech. Rep. 99/007, CISUC, 1999, http://www.mat.uc.pt/˜marta/Publicacoes/mps2.ps

[7] E. Martins, M. Pascoal, and J. Santos, "Deviation algorithms for ranking shortest paths", *Int. J. Found. Comput. Sci.*, vol. 10, no. 3, pp. 247–263, 1999.

[8] H. T. Mouftah and P.-H. Ho, *Optical Networks – Architecture and Survivability*. Boston [etc.]: Kluwer, 2003.

[9] P. Papadimitratos, Z. J. Hass, and E. G. Sirer, "Path selection in mobile ad hoc networks", in *Proc. 3rd ACM Int. Symp. Mob. Ad Hoc Netw. Comput.*, Lausanne, Switzerland, 2002, pp. 1–11.

[10] J. W. Suurballe, "Disjoint paths in networks", *Networks*, vol. 4, pp. 125–145, 1974.

[11] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths", *Networks*, vol. 14, no. 2, pp. 325–336, 1984.

Teresa Gomes received her Ph.D. degree in electrical engineering (telecommunications and electronics) from the University of Coimbra, in 1998. She is an Assistant Professor at the Department of Electrical and Computer Engineering, University of Coimbra, and a researcher at INESC Coimbra. Her research areas include reliability/quality of service analysis of telecommunications networks, teletraffic engineering and simulation.

e-mail: teresa@deec.uc.pt
Department of Electrical and Computer Engineering
Pólo II of Coimbra University
3030-290 Coimbra, Portugal

INESC Coimbra
Rua Antero de Quental 199
000-033 Coimbra, Portugal

José Craveirinha received his M.Sc. (1981) and Ph.D. degrees in E.E.S. at the University of Essex (UK) (7/1984) and the tittle of "Agregacao" in E.E.S. Telecommunications at the University of Coimbra (7/1996). He is a Full Professor at the Department of Electrical and Computer Engineering, of the University of Coimbra and has coordinated a research group in Teletraffic Theory & Network Planning at INESC-Coimbra R&D institute since 1986. He was a Director of this institute in 1994–1999. His main scientific areas of research have been stochastic modeling of teletraffic, reliability analysis and planning of telecommunication networks. His main present interests are in traffic modeling and routing in Internet and multiple objective routing in multiservice networks.
e-mail: jcrav@deec.uc.pt
Department of Electrical and Computer Engineering
Pólo II of Coimbra University
3030-290 Coimbra, Portugal

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

Artur Violante graduated in electrical and computer engineering from the University of Coimbra. He has been working since the September 2005 at Siemens SA in Alfragide Portugal, where is part of the team R-NCC Europe I.

e-mail: a.violante@gmail.com
Department of Electrical and Computer Engineering
Pólo II of Coimbra University
3030-290 Coimbra, Portugal