An exact lexicographic approach for the maximally risk-disjoint/minimal cost path pair problem in telecommunication networks

Marta Pascoal[1,2,3]* José Craveirinha[2], João Clímaco[2]

[1] University of Coimbra, CMUC, Department of Mathematics, 3001-501 Coimbra, Portugal
E-mail: marta@mat.uc.pt

[2] Institute for Systems Engineering and Computers at Coimbra, Universidade de Coimbra, rua Sílvio Lima, Pólo II, 3030-290 Coimbra, Portugal
E-mail: {jcrav, jclimaco}@inescc.pt

[3] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci, 32, Milan, 20133, Italy

20 August 2020

Abstract: The paper addresses the lexicographically maximal risk-disjoint/minimal cost path pair problem that aims at finding a pair of paths between two given nodes, which is the shortest (in terms of cost) among those that have the fewest risks in common. This problem is of particular importance in telecommunication network design, namely concerning resilient routing models where both a primary and a backup path have to be calculated to minimize the risk of failure of a connection between origin and terminal nodes, in case of failure along the primary path and where bandwidth routing costs should also be minimized. An exact combinatorial algorithm is proposed for solving this problem which combines a path ranking method and a path labelling algorithm. Also an integer linear programming (ILP) formulation is shown for comparison purposes. After a theoretical justification of the algorithm foundations, this is described and tested, together with the ILP procedure, for a set of reference networks in telecommunications, considering randomly generated risks, associated with Shared Risk Link Groups (SRLGs), and arc costs. Both methods were capable of solving the problem instances in relatively short times and, in general, the proposed algorithm was clearly faster than the ILP formulation excepting for the networks with the greatest dimension and connectivity.

Keywords: Lexicographic shortest paths; Ranking; Pairs of paths; Telecommunication network design; Resilient routing models.

1 Introduction

Multicriteria shortest path problems have important applications in telecommunication networks, specially in network routing design. Overviews on multicriteria shortest path algorithms with applications in this domain, were presented in Clímaco et al. (2016); Clímaco and Pascoal (2012). State of art reviews, focusing on MCDA (Multicriteria Decision Analysis) modelling approaches, algorithms and their applications in network design, including routing problems, can

*Corresponding author

be seen in Clímaco et al. (2016) and, in a broader context, in Clímaco and Craveirinha (2019). A particular class of these problems, with great interest in the context of resilient routing design (see a generic monography on this broad subject in Rak (2015)) involves, typically, the calculation of a pair of paths (corresponding to end-to-end routes), the primary or active path (AP) (that carries the corresponding traffic flow under normal operating conditions) and the backup or protection path (BP) (which is the path that carries that traffic when some failure affects the AP). The two paths have to be computed and signaled, for each pair of origin-destination nodes, so that the availability of the services supported by the pair may be guaranteed, as far as possible, in the event of failures. This type of problems is of paramount importance having in mind that very high levels of service availability (expressed through Service Levels Agreements for different classes of connection demands) should be maintained in the event of failures and the enormous amounts of traffic that can be lost in the event of failures in the physical or logical network structures, resulting for example from optical fiber cuts, switch/router or software failures.

In the design of routing mechanisms with built-in survivability objectives, taking into account the multi-layered structure of telecommunication networks, the concept of shared risk link group (SRLG) is frequently used, which may be defined as a group of logical links (arcs of the functional network graph representation) which share a common risk of failure. Usually the network designer, based on the information about the SRLGs associated with the arcs, seeks to calculate a pair of paths which are SRLG-disjoint, ensuring that no single fault of the AP will affect the BP, a NP-complete problem as shown in Hu (2003). However, there may arise situations for which no SRLG-disjoint path pair can be calculated, a case in which the aim of the routing procedure may consist of finding a maximally SRLG-disjoint path pair, that is a path pair with the minimal number of common SRLGs, so as to minimize the risk of simultaneous failure of the two paths. Moreover, a key concern is bandwidth usage optimization, seeking to optimize the use of bandwidth resources throughout the network links, in order to achieve the maximal possible network traffic carrying capability. This is usually represented in terms of different labels associated with the arcs of the network, representing the different risks, as well as additive path cost functions, such that the cost of using a link is some function of its capacity and used bandwidth. These considerations lead to a typical formulation of the routing problem with path protection involving the lexicographic calculation of a pair of paths which are maximally label disjoint, ideally with no label common to the AP and BP and, as a secondary objective, minimal total cost.

Several heuristic algorithms for seeking totally SRLG-disjoint path pairs have been proposed the performance of which, in terms of exactness, is usually evaluated by comparison with exact solutions from Integer Linear Programming formulations, for problems tested in reference

networks. Heuristics for this problem were proposed in Rostami et al. (2007); Todimala and Ramamurthy (2004); Xu et al. (2003). In Silva et al. (2011), a variant of the procedure in Hu (2003) is proposed where the candidate APs are considered in order of ascending cost and a BP with the minimum cost is calculated, leading to a final solution which is the pair with the least number of common SRLGs. Various heuristics were proposed for calculating totally SRLG-disjoint path pairs of minimal cost, namely Gomes et al. (2013a) and Gomes et al. (2013b). Also various heuristics were proposed for tackling maximally SRLG-disjoint path pairs of minimal cost lexicographic optimisation problems, considering variants of the objective functions or of the constraints and various resolution approaches. In particular, Gomes et al. (2016) presents two heuristics for tackling a lexicographic formulation of this type of problem which includes as additional objectives, of highest priority, that the paths are maximally node and arc disjoint.

In this work, we present an exact algorithm for solving the lexicographic formulation of the maximally risk-disjoint/minimal cost path pair problem. Noting that a one-to-one correspondence between SRLGs and risks, or labels, can be specified, the proposed algorithm is a lexicographic minimal label-minimal cost path pair algorithm which combines a path ranking method – where possible paths are ranked by increasing order of cost by using the ranking algorithm in Martins et al. (1999) – and a path labelling algorithm. This path labelling method finds the shortest path among those which have the minimal number of labels in common with the path fixed by the ranking. Also an Integer Linear Programming (ILP) formulation of the lexicographic problem, inspired by Hu (2003), is shown for performance comparison purposes.

Extensive experiments for evaluating the computational performance of the proposed lexicographic algorithm and the ILP formulation, considering four reference test networks used in the analysis of resilient routing design models in telecommunication networks Orlowski et al. (2010) and using various random labels and cost distributions, are presented. This will show that the algorithm performs clearly more efficiently than the ILP method excepting for the networks with the highest dimension and connectivity.

The remainder of this text is organized as follows. In Section 2 the notation and preliminary definitions are introduced. The lexicographic version of the problem, the ILP formulation, together with theoretical results used in the resolution method and the computational procedures of the algorithm (in Sub-section 3.1), as well as an illustrative example (Sub-section 3.2), are described in Section 3. At the end of Section 3, a study of the computational complexity of the proposed resolution algorithms, is presented. The computational experiments for assessing the performance of the lexicographic algorithm and of the ILP formulation in reference test networks, considering various distributions of random risks and arc costs, and the conclusions of these tests are also presented in this section (in Sub-section 3.4). Finally, conclusions on this study and further work are drawn in Section 4.

2 Definitions and notation

Let $G = (N, A)$ denote a directed network, where N is the set of n nodes and $A \subseteq N \times N$ is the set of m arcs. Given $v_1, v_r \in N$, a path from v_1 to v_r in G is a sequence $p = \langle v_1, v_2, \ldots, v_r \rangle$, where $(v_i, v_{i+1}) \in A$, for any $i = 1, \ldots, r-1$. Let $s, t \in N$ be called the source and the terminal nodes, respectively, and P denote the set of paths in G from s to t ($P \neq \emptyset$). Hereafter, the term path stands for a path with no repeated nodes.

Let L be the set of network labels (such that each label corresponds to one and only one failure risk), ensuring a one-to-one correspondance between labels and SRLGs, and $(i, j) \in A$ be an arc in the network. Then, the following parameters are associated with the arc (i, j):

- $L_{ij} = \{l_{ij}^1, \ldots, l_{ij}^k\} \subseteq L$, it consists of the set of risks which may affect (i, j), and
- $c_{ij} \in \mathbb{R}_0^+$, it represents the cost for using the arc (i, j).

Let $A_l = \{(i, j) \in A : l \in L_{ij}\} \subseteq A$ denote the set of arcs with label/risk l, which defines the SRLG with label l.

The set of arc labels and the cost for a given path $p \in P$ are defined by

$$l(p) = \bigcup_{(i,j) \in p} L_{ij} \quad \text{and} \quad c(p) = \sum_{(i,j) \in p} c_{ij},$$

respectively. Hereafter it is assumed that all cycles in the network have non-negative cost. Such notions can be extended to pairs of paths in P. Given a pair of paths $(p, q) \in P \times P$, the following parameters are defined:

- the number of labels that are common to both paths is defined by $l(p, q) = |l(p) \cap l(q)|$, and
- the pair's cost is defined by $c(p, q) = c(p) + c(q)$.

3 The lexicographic maximally risk-disjoint shortest pair of paths problem

Although the two objective functions introduced above, l and c, are both important, most formulations consider that the minimization of the number of risks shared by the two paths (hence the maximization of end to end service survivability in the event of failures) has higher priority than the minimization of the cost. For this reason, in the following the lexicographically maximal risk-disjoint shortest pair of paths (LMRDSPP) problem is considered. Firstly, a linear integer formulation for this problem is presented. Afterwards, the proposed algorithm is described after some preliminary theoretical results.

The goal of the LMRDSPP problem is to find a pair of paths linking nodes s and t, which minimizes the cost of the two paths, among those which have the minimal number of common labels. The problem is stated as

$$\begin{array}{ll} \text{minimize} & c(p_1, p_2) \\ \text{such that} & l(p_1, p_2) = l^* \\ & p_1, p_2 \in P \end{array} \tag{1}$$

where

$$l^* = \begin{array}{ll} \min & l(p_1, p_2) \\ \text{such that} & p_1, p_2 \in P \end{array}$$

In order to formulate the problem as an integer program, let us consider the pair of paths (p_1, p_2) and let it be defined by the decision variables:

$$x_{ij}^k = \begin{cases} 1 \text{ if the arc } (i,j) \text{ is in the path } p_k \\ 0 \text{ otherwise} \end{cases}$$

for any $(i, j) \in A$ and $k = 1, 2$. Let us also consider the variables:

$$v_l^k = \begin{cases} 1 \text{ if } l \text{ is a risk in path } p_k \\ 0 \text{ otherwise} \end{cases}$$

for any $l \in L$ and $k = 1, 2$, and

$$v_l = \begin{cases} 1 \text{ if risk } l \text{ is shared by the pair of paths} \\ 0 \text{ otherwise} \end{cases}$$

for any $l \in L$. The objective functions of the problem are:

$$l(p_1, p_2) = \sum_{l \in L} v_l,$$

for counting the number of shared risks of pair (p_1, p_2), and:

$$c(p_1, p_2) = \sum_{(i,j) \in A} c_{ij}(x_{ij}^1 + x_{ij}^2),$$

for computing the pair (p_1, p_2) total cost.

Inspired by Hu (2003), the problem can be formulated as

$$\text{minimize} \quad \sum_{(i,j)\in A} c_{ij}(x_{ij}^1 + x_{ij}^2) \tag{2a}$$

$$\text{such that} \quad \sum_{(i,j)\in A} x_{ij}^k - \sum_{(j,i)\in A} x_{ji}^k = \begin{cases} 1, & i = s \\ 0, & i \in \mathcal{N} - \{s,t\} \\ -1, & i = t \end{cases}, \quad i \in N, \quad k = 1,2 \tag{2b}$$

$$\sum_{(i,j)\in A_l} x_{ij}^k \leq \min\{n-1, |A_l|\} v_l^k, \quad l \in L, \quad k = 1,2 \tag{2c}$$

$$v_l^1 + v_l^2 - v_l \leq 1, \quad l \in L \tag{2d}$$

$$\sum_{l\in L} v_l = l^* \tag{2e}$$

$$x_{ij}^k \in \{0,1\}, \quad (i,j) \in A, \quad k = 1,2 \tag{2f}$$

$$v_l^k \in \{0,1\}, \quad l \in L, \quad k = 1,2 \tag{2g}$$

$$v_l \in \{0,1\}, \quad l \in L \tag{2h}$$

where

$$l^* = \min \sum_{l\in L} v_l \tag{3a}$$

$$\text{such that} \quad (2b) - (2d), (2f) - (2h)$$

The constraints (2b) are flow conservation constraints for the variables associated with each path from node s to node t. The conditions (2c) ensure that for each risk, l, and each path, p_k, an arc, (i,j), with that risk, is in the solution only if the associated risk variable is $v_l^k = 1$. These conditions also imply that the number of arcs in each path in the solution, for each risk, does not exceed neither $n-1$ nor the number of arcs with that risk in the network. Additionally, the set of constraints (2d) are used to relate the risk variables. These integer linear formulations can be replaced by linear relaxations with respect to the variables x_{ij}^k,

$$0 \leq x_{ij}^k \leq 1, \quad (i,j) \in A, \quad k = 1,2.$$

3.1 Resolution approach

The search for an optimal solution of the LMRDSPP problem can be restricted to pairs of loopless paths. This fact is proved below and will be used in the algorithm presented later on.

Proposition 3.1 *At least one solution of the LMRDSPP problem is a pair of loopless paths from s to t in G.*

Proof. Let us assume that any solution of the LMRDSPP problem contains at least one cycle, that is, that if (p, q) is a lexicographically optimal pair for the LMRDSPP problem, then either path p or path q contains a cycle. Let us also assume, with no loss of generality, that p is a loopless path, contrary to q which has the form $q = q_1 \diamond C \diamond q_2$, where C is any of its cycles, where the symbol $\diamond$ represents the concatenation of two paths. Therefore, the path $q^* = q_1 \diamond q_2$ has less loops than $q = q_1 \diamond C \diamond q_2$. If q^* is not loopless, then the reasoning can be repeated as many times as necessary to find a loopless path. Otherwise, (p, q^*) is a pair of simple paths from s to t. Because $l(q^*) \subseteq l(q)$ and because there are no negative cost cycles in the network, then $c(q^*) \leq c(q)$ holds, and therefore

$$l(p, q^*) \leq l(p, q) \quad \text{and} \quad c(p, q^*) \leq c(p, q).$$

Additionally, due to the optimality of (p, q), the pair of paths (p, q^*) is also optimal and may be a better solution, which contradicts the assumption. □

According to Proposition 3.1, it is sufficient to find pairs of loopless paths in order to solve the LMRDSPP problem. The algorithm for finding an optimal pair of paths is based on three main ideas:

- The first one is to list possible primary paths, say p, by increasing order of c.
- The second is to find the best backup path with respect to each primary path p. This best backup path is a shortest path among those which have the least number of risks in common with the path p. That is, the path can be found by solving the new problem

$$\begin{array}{ll} \text{lexmin} & (l(p, q), c(p, q)) \\ \text{such that} & q \in P \end{array} \tag{4}$$

- Additionally, an upperbound on the number of risks shared by any two paths from s to t is used, $RiskUB$. This value is updated as new pairs of paths are computed and it is used to limit the search for pairs of paths that can be optimal.

The first of these points can be addressed by applying an algorithm for ranking paths by order of their cost c, for instance one of the methods in Katoh et al. (1982); Martins et al. (1999); Yen (1971). The second is handled by means of a dynamic programming algorithm described below.

Let p be a fixed primary path from s to t and $RiskUB$ be the current number of shared risks upperbound. The proposed method generates several paths extending an initial path starting in s by adding one node at a time, for finding a path that optimizes problem (4), while respecting the upperbound $RiskUB$. Several paths from s to another node i, $i \in N$, can be found, therefore each one is identified as p_x, with x an index associated with the node i, and a label $L_x = [p_x, \beta_x, \pi_x^r, \pi_x^l, \pi_x^c]$, with the following components:

- p_x, the sequence of nodes in the path from s to x;
- β_x, the network node which corresponds to x, i.e., i;
- π_x^r, the set of risks in p which also appear in the path p_x;
- π_x^l, the number of risks in p which also appear in the current path p_x;
- π_x^c, the cost of the path p_x.

When examining a given path p_x from node s to node i, it is intended to extend it by scanning all the arcs $(i, j) \in A$, where $\beta_x = i$. Additionally, the computation of more than one path from s to i may be required, therefore different indexes are used to distinguish them. The value $\pi_x^l = |\pi_x^r|$ is just used for the sake of clarity of the presentation.

For this specific subproblem, the priority criterion is the number of risks shared by the two paths. A counter example that this criterion does not satisfy Bellman's Optimality Principle is presented in the next subsection. This fact makes the comparison between paths/labels for the same ending node harder. The dominance between two different labels L_x and L_y is defined below, in order to enable this comparison.

Definition 3.1 *Given the node i and two of its labels, L_x and L_y, such that $\beta_x = \beta_y = i$, label L_x dominates label L_y if $\pi_x^r \subseteq \pi_y^r$ and $\pi_x^c < \pi_y^c$.*

Proposition 3.2 shows that the dominated labels are of no use for finding an optimal solution of problem (1).

Proposition 3.2 *Let p be a path in P and $RiskUB \geq 0$. If L_x is a dominated label of a given node i and p_x is the corresponding path from s to i, then no path from s to t that contains p_x is an optimal solution of (4).*

Proof. Let q be any path from node i to node t, and by contradiction assume that $p_x \diamond q$ is a lexicographic optimal path from s to t. By assumption the label L_x is dominated by another label of node i, L_y, corresponding to the path p_y from s to i. Therefore, by Definition 3.1, $\pi_y^r \subseteq \pi_x^r$ and $\pi_y^c < \pi_x^c$. If $\pi_y^r \subseteq \pi_x^r$, then $\pi_y^l \leq \pi_x^l$ and also $l(p, p_y \diamond q) \leq l(p, p_x \diamond q)$. Additionally, $\pi_y^c < \pi_x^c$, thus $c(p_y \diamond q) < c(p_x \diamond q)$ and $c(p, p_y \diamond q) < c(p, p_x \diamond q)$. Therefore, $p_y \diamond q$ is better than $p_x \diamond q$, so $p_x \diamond q$ could not be optimal. This leads to a contradiction, which concludes the proof. □

The algorithm starts with a path formed simply by the initial node, s, which is extended depending on on the arcs that emerge from node s and on the comparison of each extension

with paths previously found. The new extensions are stored as new labels to be scanned. The analysis of a new label L_x, associated with a path p_x from node s to node $\beta_x = i$ is similar to that of node s. Any arc $(i, j) \in A$ is considered, associated with the label $L_y = [p_y, \beta_y, \pi^r_y, \pi^l_y, \pi^c_y]$, where:

- $p_y = p_x \diamond \langle i, j \rangle$;
- $\beta_y = j$;
- $\pi^r_y = \pi^r_x \cup L_{ij} \cap l(p)$;
- $\pi^l_y = |\pi^r_y|$;
- $\pi^c_y = \pi^c_x + c_{ij}$;

provided that it does not exceed the current best number of shared risks and it is not dominated by any other label already established for node j. Each label that is obtained corresponds to a path between the nodes s and j, which may be part of an optimal path. Therefore, it is compared to other paths previously generated and under the same conditions, so that they can be discarded as early as possible in case they are dominated. Additionaly, when node $j = t$, the minimum number of shared risks and the current optimal path are updated. The newly generated labels are stored until they are scanned.

Given a path, p, and an upperbound on the number of shared risks, *RiskUB*, Algorithm 1 outlines the steps for finding a shortest path from s to t among those which have the fewest risks in common with p. The goal of the method is to create a tree, rooted at node s, which contains paths from s to any node i that correspond to non-dominated labels according to the definition above. The tree is extended by scanning its nodes and considering the arcs emerging from each of them. The control of the growth of the tree is based on two results. On the one hand Proposition 3.2 is used to restrict the new labels that are created, as well as it allows to discard some of the already created ones. On the other hand, the value *RiskUB* is updated every time the best stored path from s to t is improved with respect to the number of shared risks, and partial paths that lead to solutions worse than that one are never taken into account. The dominance tests between labels on lines 14 and 17 in Algorithm 1 are implemented by pairwise comparison of labels with respect to their sets of shared risks, their cardinality and the corresponding costs. Hence, by using this form of "lazy evaluation" of the comparison between labels, the label costs of two labels are only compared if one of the risks set is contained in the other.

Algorithm 1 is combined with a general procedure for solving the LMRDSPP problem. In this framework the primary paths are ranked by order of cost and the potential shortest

backup path with the fewest labels in common with each of them is computed when calling Algorithm 1. The best pair of paths is stored in the variable $(BestP, BestQ)$, which is updated after calling Algorithm 1 whenever the solution is improved. Like before $RiskUB$ is another auxiliary variable, used to store the best number of shared risks found so far by the method. This variable is updated in Algorithm 1, before the pair $(BestP, BestQ)$ is also changed. The method is outlined in Algorithm 2 – designated hereafter as `SLA` (Single criterion Lexicographic Algorithm).

The variable $RiskUB$ stores an upperbound on the optimal number of shared risks. This bound is improved whenever a pair of paths with fewer common risks than the current value is generated, according to line 21 of Algorithm 1. Therefore, the following results holds.

Lemma 3.1 *The sequence $\{l(p_k, q)\}_{k\geq 1}$, where (p_k, q) are the pairs of paths generated by Algorithm 2, $k = 1, 2, \ldots$, is non-increasing.*

The correctness of Algorithm 2 is proved in Proposition 3.3.

Proposition 3.3 *The pair of paths output by Algorithm 2 is a lexicographically optimal solution for the LMRDSPP problem.*

Proof. Let us assume that the solution generated by Algorithm 2, say (p^*, q^*), is not an optimal solution. That is, assume that another solution exists given by Algorithm 2, (p', q'), such that either $l(p', q') < l(p^*, q^*)$ or $l(p', q') = l(p^*, q^*)$ and $c(p', q') < c(p^*, q^*)$.

Three situations may occur when ranking paths by order of cost in Algorithm 2:

1. The path p' appears before path p^* in the ranking. Then, because the pair (p', q') is the result of an earlier call of Algorithm 1, by Lemma 3.1, $l(p', q') \geq l(p^*, q^*)$, and because by assumption (p', q') dominates (p^*, q^*), $l(p', q') = l(p^*, q^*)$ must hold. Additionally, $c(p', q') < c(p^*, q^*)$, and thus the pair (p^*, q^*) could not have been computed by Algorithm 1 nor be the output of Algorithm 2.

2. If $p' = p^*$, then the pair of paths (p^*, q^*) could not have been computed by Algorithm 1, given that it is dominated by the pair (p', q').

3. The path p' appears after path p^* in the ranking. Then, the result of Algorithm 1 when considering p' is either the pair (p', q') or another one, (p', q''), which is lexicographically better than (p', q'). By assumption the pair (p', q') dominates (p^*, q^*), therefore either (p', q') or (p', q'') replaces the previous pair as the best solution and (p^*, q^*) could not be the solution given by Algorithm 2.

In either case the pair (p^*, q^*) cannot be the solution given by Algorithm 2, as initially assumed. □

3.2 Example

As an example for the LMRDSPP problem, let us consider the network G_1 in Figure 1a with unitary arc costs, $c_{ij} = 1$, and the sets L_{ij} defined by the different arc colors (or different letters in the figures), for any $(i, j) \in A$. The full list of pairs of paths linking $s = 1$ to $t = 4$ in this network is shown in Table 1 without repetition of the same pair of paths in reverse order. It is worth noting that the algorithm does not prevent the generation of two equal paths in the pair, even though such a solution would be useless in practice. The optimal solution for the LMRDSPP problem is the pair of paths $(\langle 1, 3, 4\rangle, \langle 1, 3, 2, 4\rangle)$, with one shared risk (the green one) and cost 5.

(a) Network G_1 (b) Network G_2

Figure 1: Example networks

As a counter example that shows that Bellman's Optimality Principle does not hold when finding a shortest backup path with at most a given number of shared risks, consider the network G_2 in Figure 1b and the primary path $p = \langle 1, 3, 4\rangle$, with the risks $l(p) = \{r, g, b\}$. Given the labels associated with node 2

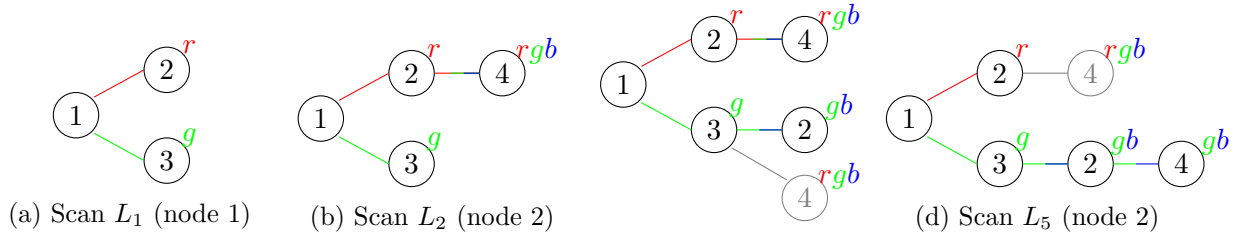
$$L_x = (\langle 1, 2\rangle, 2, \{r\}, 1, 1) \text{ and } L_y = (\langle 1, 3, 2\rangle, 2, \{g, b\}, 2, 2),$$

the first is better than the latter in the sense that $\pi^l_x = 1 < \pi^1_y = 2$ and $\pi^c_x = 1 < \pi^c_y = 2$. However, after extending each of these paths to node 4 by adding the arc $(2, 4)$, the paths identified by

$$L_{x'} = (\langle 1, 2, 4\rangle, 4, \{r, g, b\}, 3, 2) \text{ and } L_{y'} = (\langle 1, 3, 2, 4\rangle, 4, \{g, b\}, 2, 3)$$

are obtained and, in this case, the latter label is better than the first because $\pi^l_{x'} = 3 > \pi^1_{y'} = 2$. This means that two partial labels for the same node cannot be compared directly with respect to the number of risks shared with the primary path, given that the objective function values of their extensions depend on the risks of all the arcs.

Instead, when applying Algorithm 1 to the network G_2 depicted in Figure 1b under the above conditions, with $p = \langle 1, 3, 4\rangle$ as the primary path and at most $RiskUB = 3$ risks shared with path p, the trees in Figure 2 and the labels listed in Table 2 are found. Then, the best pair of computed paths is $(\langle 1, 3, 4\rangle, \langle 1, 3, 2, 4\rangle)$, which has 2 shared risks and cost 5, as explained next in detail.



(a) Scan L_1 (node 1) (b) Scan L_2 (node 2) (c) Scan L_3 (node 3) (d) Scan L_5 (node 2)

Figure 2: Paths from 1 to 4 in network G_2 with at most 3 risks in common with $\langle 1, 3, 4\rangle$

The first label to consider is L_1, for the initial node 1. Because two arcs emerge from node 1, two labels are created when the node corresponding to L_1 is considered: one associated with node 2, that is, path $\langle 1, 2\rangle$, and another one associated with node 3, that is, path $\langle 1, 3\rangle$ – Figure 2a. Assuming that L_2 is the next label to be scanned, the path $\langle 1, 2, 4\rangle$ is obtained, associated with the label L_4 – Figure 2b. Similarly, when scanning label L_3, the paths $\langle 1, 3, 2\rangle$ and $\langle 1, 3, 4\rangle$ are obtained – Figure 2c. The first cannot be compared to the other path until node 2, $\langle 1, 2\rangle$, because their sets of risks are not contained in one another, therefore it is stored and associated with the label L_5. However, the second corresponds to a path until the terminal node with exactly the same shared risks and cost as $\langle 1, 2, 4\rangle$. Because it is not better than the former, it is discarded. The next label to scan is L_5, for path $\langle 1, 3, 2\rangle$, which can be extended to $\langle 1, 3, 2, 4\rangle$ – Figure 2d. This path is also compared to the previous path to node 4, $\langle 1, 2, 4\rangle$. The risks shared by the new secondary path are included in those shared by the former and are only 2, thus it is concluded that the former secondary path is dominated by the new one and $RiskUB$ is updated with 2. No further labels are added to the search tree, therefore the best pair of computed paths is $(\langle 1, 3, 4\rangle, \langle 1, 3, 2, 4\rangle)$, which has 2 shared risks and cost 5.

When applying Algorithm 2 for finding the optimal pair of paths for the LMRDSPP problem in the network G_1 – Figure 1a, the initial upperbound is $RisksUB = \infty$ and the following paths are ranked:

- $p_1 = \langle 1, 2, 4\rangle$ and then $q = \langle 1, 3, 4\rangle$ with 2 shared risks and cost 4. Then $RiskUB$ is updated to 2 and the best pair of paths to $(BestP, BestQ) = (p_1, q)$.
- $p_2 = \langle 1, 3, 4\rangle$ and then $q = \langle 1, 3, 2, 4\rangle$ with 1 shared risk and cost 5. Then $RiskUB$ is updated to 1 and the best pair of paths is $(BestP, BestQ) = (p_2, q)$.
- $p_3 = \langle 1, 2, 3, 4\rangle$ and then $q = \langle 1, 3, 2, 4\rangle$ with 1 shared risk and cost 6.
- $p_4 = \langle 1, 3, 2, 4\rangle$ and then $q = \langle 1, 3, 4\rangle$ with 1 shared risks and cost 5.

At the end of the algorithm, the optimal pair of paths is $(\langle 1, 3, 4\rangle, \langle 1, 3, 2, 4\rangle)$.

As another example of Algorithm 1, consider its application to the network G_1 depicted in Figure 1a when the primary path from node 1 to node 4 is $p = \langle 1, 2, 4\rangle$ in G_1 (one of the shortest). Consider also that at most $RiskUB = 3$ risks are allowed to be shared with path p. Then, Algorithm 1 produces the search tree in Figure 3 and the best pair of paths when p is fixed is $(\langle 1, 2, 4\rangle, \langle 1, 3, 4\rangle)$, with 2 shared risks and cost 4.

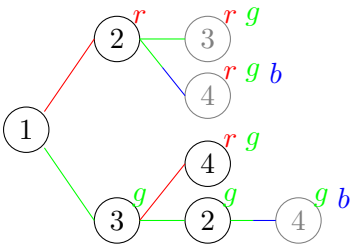


Figure 3: Paths from 1 to 4 in network G_1 with at most 3 risks in common with $\langle 1, 2, 4\rangle$

When applying Algorithm 2 for finding the optimal pair of paths for the LMRDSPP problem, the initial upperbound is $RisksUB = \infty$ and the following paths are ranked:

- $p_1 = \langle 1, 2, 4\rangle$ and then $q = \langle 1, 3, 4\rangle$ with 2 shared risks and cost 4. Then $RiskUB$ is updated to 2 and the best pair of paths to $(BestP, BestQ) = (p_1, q)$.
- $p_2 = \langle 1, 3, 4\rangle$ and then $q = \langle 1, 3, 2, 4\rangle$ with 1 shared risk and cost 5. Then $RiskUB$ is updated to 1 and the best pair of paths is $(BestP, BestQ) = (p_2, q)$.
- $p_3 = \langle 1, 2, 3, 4\rangle$ and then $q = \langle 1, 3, 2, 4\rangle$ with 1 shared risk and cost 6.
- $p_4 = \langle 1, 3, 2, 4\rangle$ and then $q = \langle 1, 3, 4\rangle$ with 1 shared risks and cost 5.

At the end of the algorithm, the optimal pair of paths is $(\langle 1, 3, 4\rangle, \langle 1, 3, 2, 4\rangle)$.

3.3 Computational complexity

This section is devoted to the estimation of the computational complexity order of the presented method. The method consists of two components: paths ranking (in Algorithm 1) and dynamic programming for finding a shortest path among those with the fewest risks in common with a path p (Algorithm 2).

Different ranking algorithms can be applied, but the time or the number of operations executed by Algorithm 2 depends on the total number of paths ranked until the solution is found. For instance, if only loopless paths are computed, Yen's algorithm can be used, with time of $O(m + n \log n + K_1 n\, g(m, n))$ Yen (1971). Otherwise, if it is allowed to compute (and discard) paths containing loops, the algorithm by Martins, Pascoal and Santos can be used, with

$O(m+n\log n+K_2 g(m,n,|L|))$ Martins et al. (1999). Here $O(m+n\log n)$ is the time complexity order for finding a shortest path, $g(m,n,|L|)$ represents the number of operations required by Algorithm 2, and K_1 (K_2) stands for the number of loopless paths (paths) analyzed by each algorithm. If an upper bound for these numbers is not set in advance, as shown in Algorithm 2, then all the loopless paths (paths) from node s to node t are listed.

The number of labels of a given node created by Algorithm 1 is at most the number of possible paths from node s to that node. Considering that these paths may be any sorted sequence of nodes with between 2 and n nodes, that number of paths is given by $\sum_{k=2}^{n} P(n,k-2)$, where $P(n,k)$ denotes the number of k-permutations of n nodes, and thus $\sum_{k=2}^{n} P(n,k-2) \leq n^{n-1}$. Moreover, scanning each label implies creating at most n new more labels. Creating a label has time of $O(|L|+2)$, and testing its dominance is done by comparing it with previous labels, which requires at most $\sum_{k=1}^{|L|} \binom{|L|}{k} = 2^{|L|}$ comparisons, and thus it has time of $O(2^{|L|})$. Therefore, Algorithm 1 is of $O(n^{n-1}(|L|+2^{|L|}))$, or simply $O(n^{n-1}2^{|L|})$.

Proposition 3.4 *The worst-case number of operations performed by Algorithm 1 is of $O(n^{n-1}2^{|L|})$, and Algorithm 2 is of $O(m+n\log n+K\,n^{n+1}2^{|L|})$ or of $O(m+n\log n+K\,n^{n}2^{|L|})$, respectively, if K paths are ranked with the Yen's algorithm, or with MPS algorithm, respectively.*

3.4 Computational experiments

Computational tests were run to assess the performance of the method introduced earlier as well as to compare it with the mixed integer formulation given by (2) – (3). With this purpose, Algorithm 2 (denoted by `SLA`) was coded in C language. In order to rank the simple paths in G by order of cost, the code `SLA` uses the MPS algorithm, Martins et al. (1999). A maximal number of 7×10^6 generated paths was imposed in the code. Additionally, the formulation (2) – (3), hereafter designated as `ILP`, was solved with CPLEX 12.7. The imposition, in Algorithm 1, of that bound on the number of generated paths is associated with computer memory requirements and guarantees that, in the vast majority of the application ranges of the resolution method, an optimal solution may be obtained. This will be shown in the experimental results described next. This is a common procedure when combinatorial algorithms of this type are applied to telecommunication networks and also for performance comparison with the ILP solutions in terms of the resulting CPU times. As a consequence, there will be a few cases for which that bound is attained so that the algorithm stops and only sub-optimal solutions are obtained. This

will be illustrated in some results for the network `Germany50`, the one of greater dimension and connectivity in the experimental setting.

All tests ran on an Intel® i7-6700 Quad core, with 8Mb of cache, a 3.4 GHz processor and 16 Gb of RAM, over openSUSE Leap 42.2

3.4.1 Test bed

The set of experiments used reference networks from the literature in telecommunications, mentioned in Betker et al. (2003); Orlowski et al. (2010) and summarized in Table 3. These include the network `Cost266`, originated from the project COST266-Advanced Infrastructure for Photonic Networks Maesschalck et al. (2003) of the European Cooperation in the Field of Scientific and Technical Research. It also includes `NobelEU` and `Germany50`, reference networks originated from the European project NOBEL (NOBEL, 2019). These networks are undirected, therefore each of their undirected arcs, $\{i, j\}$, was duplicated as two directed arcs in opposite directions, (i, j) and (j, i). The values n and m in Table 3 refer to the number of nodes and arcs in the used network representation.

The results presented in the following are mean values obtained for 10 different seeds and 45 origin-destination pairs, that is, 450 instances, for each set of parameters. For each arc $(i, j) \in A$, the cost c_{ij} represents link occupation and is given by $c_{ij} = 1/b_{ij}$, where the available bandwidths b_{ij} are randomly generated, according to the distributions shown in Table 4, in the sets:

$$\begin{aligned} I_i &= \{2 + 2k : k = 20i, \ldots, 20(i + 1) - 1\} \quad (i = 0, 1, 2) \\ I_3 &= \{2 + 2k : k = 60, \ldots, 78\} \end{aligned}$$

The distributions D1, D2 and D3 represent uniformly, highly and lightly loaded networks, respectively. The SRLGs L_{ij} are uniformly generated between 1 and $|L| = 15, 20, 25$, with mean number of SRLGs per arc $\alpha = 1, 2, 4$

3.4.2 Test results

The average run times (in seconds) for each set of parameters are shown in Figures 4–8. In the case of the `Germany50` network, the code `SLA` was halted when the memory space mentioned earlier was exceeded, which in this case happened for all the instances. For this reason, although both the run times obtained by `ILP` and by `SLA` are presented in Figure 8, only the first ones correspond to problems that ran until the end.

For the first four types of networks the two approaches tend to perform slower as α increased. This behavior was more consistent for the code `SLA` and the increase in the CPU time can be

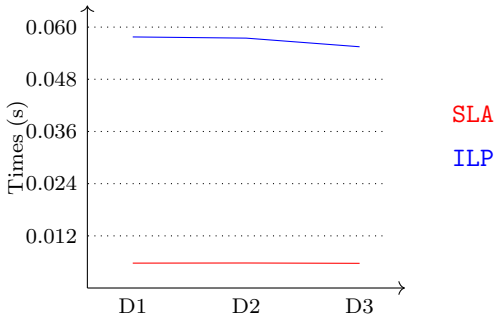


Figure 4: Mean CPU time in `NSFFixedLabels` network

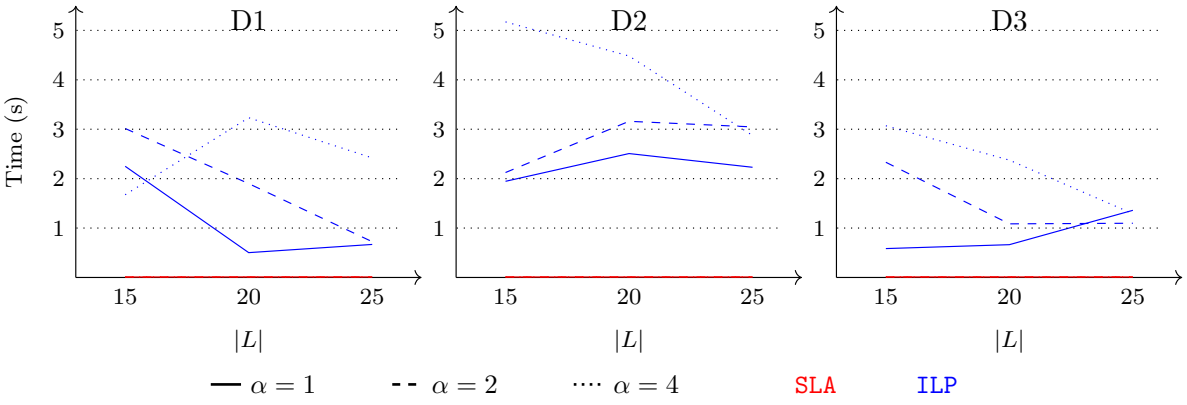


Figure 5: Mean CPU time in `NSFRandomLabels` networks

explained by the more demanding comparison between node labels when the number of risks is bigger. The CPU times for the `ILP` were always smaller than 6 seconds. In the case of code `SLA`, the CPU times slightly increased with the size of the network, but were in general clearly less than 1 second, excepting for the `Germany50` network – Figure 8. It is important to note that the proposed algorithm clearly outperformed the `ILP` procedure for all networks/instances excepting in a particular situation for the `Cost266` networks with distribution D2, $|L| = 25$ and $\alpha = 4$ – Figure 7 – and for the larger network, `Germany50`.

As mentioned earlier, the times of the code `SLA` shown in Figure 8 correspond to the mean run times until the algorithm was halted due to the required memory space. The best solution found by the algorithm was compared to the optimal solution obtained by the linear integer formulation `ILP` for the `Germany50` networks. The percentage of instances for which the code `SLA` was capable of finding an optimal solution is reported in Table 5. According to these results the optimal solution was found for most cases, and in the worst case the optimum could not be found for 7% of the instances for distribution D3, with $|L| = 15$ and $\alpha = 1$. Table 6 summarizes the results obtained for three instances of the `Germany50` networks, with $|L| = 15$ labels and $\alpha = 1$, for which `SLA` was not capable of finding an optimal solution. For these

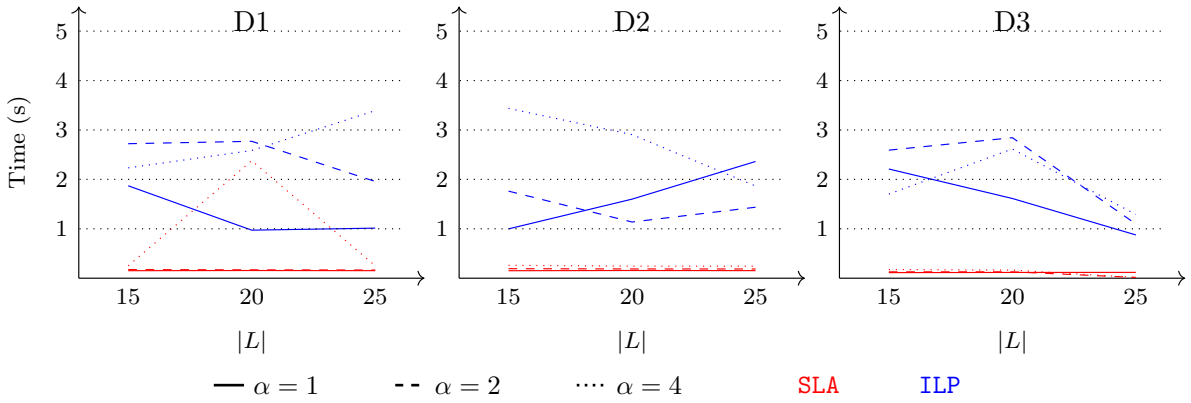


Figure 6: Mean CPU time in `NobelEU` networks

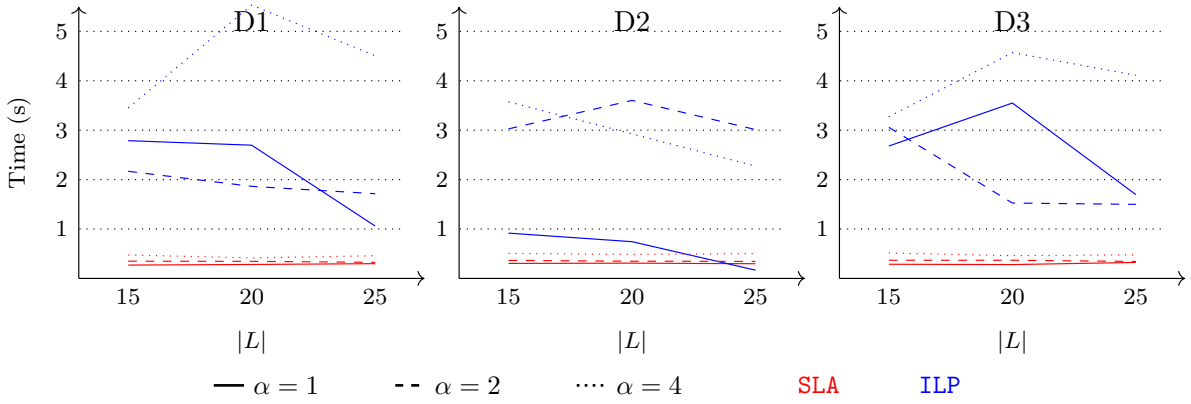


Figure 7: Mean CPU time in `Cost266` networks

instances the obtained sub-optimal solutions include path pairs with one risk in common while the optimal ones are risk-disjoint but have higher costs. Nevertheless, these were very rare cases, in the considered experimentation setting, further noting that for all the instances, in the other networks, the computed solutions were always optimal.

The CPU times of the proposed resolution method are fully compatible with resilient routing operational design involving off-line calculation of pairs of primary and back-up protection paths in various types of telecommunication transport networks such as optical networks or MPLS-TP (Multiprotocol Label Switching-Transport Profile) networks. They are even compatible with dynamic end-to-end protection mechanisms (for non real-time application) with up-dating periods of not less than ten seconds for all typical network scenarios.

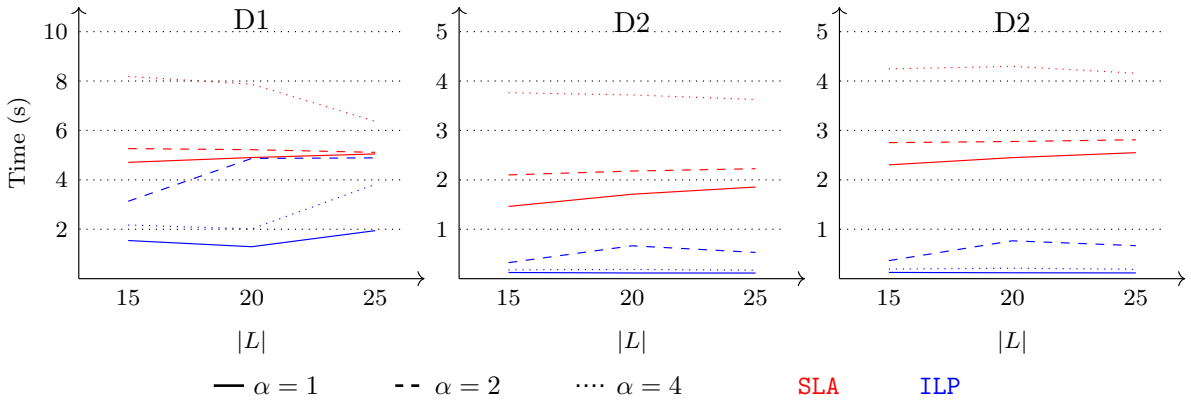


Figure 8: Mean CPU time in `Germany50` networks

4 Conclusions

We presented an exact algorithm for solving the lexicographic maximally risk-disjoint/minimal cost path pair problem. This is the most common formulation of a routing design problem in telecommunication networks, involving the joint calculation of an active and a back-up path for each node to node connection, both paths being subject to failures represented through SRLGs. The proposed resolution method is a lexicographic minimal label-minimal cost path pair algorithm which combines a path ranking method and a path labelling algorithm. Also, an Integer Linear Programming (ILP) formulation of this lexicographic problem, inspired by Hu (2003), was considered for performance comparison purposes.

Extensive experiments for evaluating the computational performance of the proposed lexicographic algorithm and the ILP formulation, applied to four reference test networks (commonly used in the analysis of resilient routing design models in telecommunication networks) and using various random labels and cost distributions, were carried out. These experiments have shown that the algorithm performs clearly more efficiently than the ILP method excepting for the networks with the highest dimension and connectivity. The computational code ran from a few ms to a few hundreds of ms in networks of low or medium size/connectivity and up to some seconds for the greater networks. In general, the CPU times increased with the size of networks, the connectivity and, in most cases, with the increase in the average number of SRLGs/labels per arc.

These results make the proposed algorithm suitable for a wide range of applications in telecommunication resilient routing design. Even for the larger networks, for which is less efficient than the ILP formulation, the algorithm may still be useful for practical or economic reasons since it does not require the installing of CPLEX routines in all routers (or path computational elements) of the network with the inherent computational requirements and licensing

costs for the network operator. Finally, a possible adaptation of the core algorithms to the bicriteria optimisation version of the maximally risk-disjoint/minimal cost path pair problem, deserves further investigation.

Acknowledgments This work was partially supported by the Portuguese Foundation for Science and Technology (FCT) under project grants UID/MAT/00324/2020 and UIDB/00308/2020. The work was also partially financially supported by project P2020 SAICTPAC/0011/2015, co-financed by COMPETE 2020, Portugal 2020 - Operational Program for Competitiveness and Internationalization (POCI), European Union's European Regional Development Fund, and FCT, and by FEDER Funds and National Funds under project CENTRO-01-0145-FEDER-029312. We also wish to thank Dr. Teresa Gomes (INESC Coimbra - University of Coimbra) for her helpful suggestions concerning the test networks and experimental settings.

References

Betker, A., Gerlach, C., Hülsermann, R., Jäger, M., Barry, M., Bodamer, S., Späth, J., Gauger, C., and Köhn, M. (2003). Reference transport network scenarios. Technical report.

Clímaco, J. and Craveirinha, J. (2019). *MCDA/M in Telecommunication Networks: Challenges and Trends*, pages 11–55. Chapman and Hall/CRC, New York, NY.

Clímaco, J., Craveirinha, J., and Girão-Silva, R. (2016). *Multicriteria analysis in telecommunication network planning and design: A survey*, volume 233, pages 1167–1233. Springer, New York, NY.

Clímaco, J. and Pascoal, M. (2012). Multicriteria path and tree problems: discussion on exact algorithms and applications. *International Transactions in Operational Research*, 19:63–98.

Gomes, T., Jorge, L., Melo, P., and Girão-Silva, R. (2016). Maximally node and SRLG-disjoint path pair of min-sum cost in GMPLS networks: a lexicographic approach. *Photonic Network Communications*, 31(1):11–22.

Gomes, T., Jorge, L., Melo, P., Girão-Silva, R., and Mendes, S. (2013a). Calculating a maximally node and SRLG-disjoint path pair of min-sum cost in GMPLS networks. In *9th International Conference on the Design of Reliable Communication Networks (DRCN 2013)*, pages 314–321.

Gomes, T., Simões, C., and Fernandes, L. (2013b). Resilient routing in optical networks using SRLG-disjoint path pairs of min-sum cost. *Telecommunication Systems Journal*, 52(2):737–749.

Hu, J. Q. (2003). Diverse routing in optical mesh networks. *IEEE Transactions on Communications*, 51(3):489–494.

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

Maesschalck, S. D., Colle, D., Lievens, I., Pickavet, M., Demeester, P., Mauz, C., Jäger, M., Inkret, R., Mikac, B., and Derkacz, J. (2003). Pan-european optical transport networks: An availability-based comparison. *Photonic Network Communications*, 5:203–225.

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

NOBEL (Accessed in November 25, 2019). Next generation optical networks for broadband european leadership. *Available at http://www.ist-nobel.org*.

Orlowski, S., M., R. W., Pióro, and Tomaszewski, A. (2010). SNDlib 1.0 – survivable network design library. *Networks*, 55(3):276–286.

Rak, J. (2015). *Resilient routing in communication networks*. Springer.

Rostami, M., Khorsandi, S., and Khodaparast, A. (2007). Cose: A SRLG-disjoint routing algorithm. In *Fourth European Conference on Universal Multiservice Networks (ECUMN'07)*, pages 86–92.

Silva, J., Gomes, T., Fernandes, L., Simões, C., and Craveirinha, J. (2011). An heuristic for maximally SRLG-disjoint path pairs calculation. In *2011 3rd International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 1–8.

Todimala, A. and Ramamurthy, B. (2004). IMSH: An iterative heuristic for SRLG diverse routing in wdm mesh networks. In *Proceedings 13th International Conference on Computer Communications and Networks (ICCCN 2004)*, pages 199–204. IEEE.

Xu, D., Xiong, Y., Qiao, C., and Li, G. (2003). Trap avoidance and protection schemes in networks with shared risk link groups. *Journal of Lightwave Technology*, 21(11):2683–2693.

Yen, J. (1971). Finding the K shortest loopless paths in a network. *Management Science*, 17:712–716.

Algorithm 1: Shortest path among those with the fewest risks in common with path p (less than $RiskUB$)

input: Path p
Upperbound $RiskUB$
Pair of paths $(BestP, BestQ)$

output: Path q, the shortest among those with the fewest risks in common with p (less than $RiskUB$)
Updated value of $RiskUB$

$nX \leftarrow 1$
Insert label $L_{nX} = [\langle 1 \rangle, s, \emptyset, 0, 0]$ in X
$q \leftarrow -$
$BestC \leftarrow c(BestP, BestQ)$
while $X \neq \emptyset$ **do**
  $x \leftarrow$ element in X
  $X \longleftarrow X - \{x\}$
  $i \leftarrow \beta_x$
  for *all* $(i, j) \in A$ **do**
    $\delta^r \leftarrow \pi^r_x \cup L_{ij} \cap l(p)$
    $\delta^l \leftarrow |\delta^r_j|$
    $\delta^c \leftarrow \pi^c_x + c_{ij}$
    if $\delta^l \leq RiskUB$ **then**
      if $(\delta^r, \delta^l, \delta^c)$ *is not dominated by any other label of* j **then**
        $nX \leftarrow nX + 1$
        Insert $L_{nX} = [p_x \diamond \langle x, nX \rangle, j, \delta^r, \delta^l, \delta^c]$ in X
        Eliminate other labels of j dominated by $(\delta^r, \delta^l, \delta^c)$
        if $j = t$ **then**
          if $\delta^l < RiskUB$ **then**
            $q \leftarrow p_{nX}$
            $RiskUB \leftarrow \delta^l$
            $BestC \leftarrow \delta^c$
          else if $\delta^l = RiskUB$ *and* $\delta^c < BestC$ **then**
            $q \leftarrow p_{nX}$
            $BestC \leftarrow \delta^c$

Algorithm 2: Finding an optimal pair of paths for the LMRDSPP problem

$(BestP, BestQ) \leftarrow (-, -)$
$RiskUB \leftarrow \infty$
for $k = 1, 2, \ldots$ **do**
    $p_k \leftarrow k$-th shortest path from s to t
    $(p_k, q) \leftarrow$ Shortest path among those with the fewest risks in common with p_k (with bound $RiskUB$), obtained by Algorithm 1
    if $RiskUB < l((BestP, BestQ))$ *or* $(RiskUB = l(BestP, BestQ)$ *and* $c(p_k, q) < c(BestP, BestQ))$ **then**
        $(BestP, BestQ) \leftarrow (p_k, q)$

Table 1: Pairs of paths from 1 to 4 in the network G_1

Path p_k	Path q	$l(p_k) \cap l(q)$	$l(p_k, q)$	$c(p_k, q)$
$p_1 = \langle 1, 2, 4 \rangle$	$\langle 1, 2, 4 \rangle$	$\{r, g, b\}$	3	4
	$\langle 1, 3, 4 \rangle$	$\{r, g\}$	2	4
	$\langle 1, 2, 3, 4 \rangle$	$\{r, g\}$	2	5
	$\langle 1, 3, 2, 4 \rangle$	$\{g, b\}$	2	5
$p_2 = \langle 1, 3, 4 \rangle$	$\langle 1, 3, 4 \rangle$	$\{r, g\}$	2	4
	$\langle 1, 2, 3, 4 \rangle$	$\{r, g\}$	2	5
	$\langle 1, 3, 2, 4 \rangle$	$\{g\}$	1	5
$p_3 = \langle 1, 2, 3, 4 \rangle$	$\langle 1, 2, 3, 4 \rangle$	$\{r, g\}$	2	6
	$\langle 1, 3, 2, 4 \rangle$	$\{g\}$	1	6
$p_4 = \langle 1, 3, 2, 4 \rangle$	$\langle 1, 3, 2, 4 \rangle$	$\{g, b\}$	2	6

Table 2: Paths from 1 to 4 in network G_2 with at most 3 risks in common with $\langle 1, 3, 4 \rangle$

L_{nX}	p_x	j	π_x^r	π_x^l	π_x^c
L_1	$\langle 1 \rangle$	1	$\emptyset$	0	0
L_2	$\langle 1, 2 \rangle$	2	$\{r\}$	1	1
L_3	$\langle 1, 3 \rangle$	3	$\{g\}$	1	1
L_4	$\langle 1, 2, 4 \rangle$	4	$\{r, g, b\}$	3	2
L_5	$\langle 1, 3, 2 \rangle$	2	$\{g, b\}$	2	2
L_6	$\langle 1, 3, 2, 4 \rangle$	4	$\{g, b\}$	2	3

Table 3: Test parameters

Network	n	m	$\delta = m/n$	$\lvert L \rvert$	α
`NSFFixedLabels` Betker et al. (2003)	11	52	4.7	21	1.5
`NSFRandomLabels` Betker et al. (2003)	14	42	3.0	15, 20, 25	1, 2, 4
`NobelEU` Orlowski et al. (2010)	28	80	2.9	15, 20, 25	1, 2, 4
`Cost266` Orlowski et al. (2010)	37	114	3.1	15, 20, 25	1, 2, 4
`Germany50` Orlowski et al. (2010)	50	176	3.5	15, 20, 25	1, 2, 4

Table 4: Available bandwidth distributions

	I_0	I_1	I_2	I_3
D1	25%	25%	25%	25%
D2	70%	15%	10%	5%
D3	18%	18%	18%	46%

Table 5: Problems for which `SLA` found all optimal solutions (%) in the `Germany50` network

	$\alpha = 1$			$\alpha = 2$			$\alpha = 4$		
Dist.$\backslash\|L\|$	15	20	25	15	20	25	15	20	25
D1	100	100	100	99	100	100	100	100	98
D2	100	100	100	100	100	100	100	100	99
D3	93	100	100	100	99	100	100	100	99

Table 6: Instances for which `SLA` did not obtain the optimal solutions in the `Germany50` network

CPU time (ms)	Best (l, c) found by `SLA`	Optimal (l^*, c^*)
2442.225	$(1, 1416.880)$	$(0, 2396.6318)$
2616.893	$(1, 1381.037)$	$(0, 2396.6318)$
2827.702	$(1, 1444.887)$	$(0, 2760.8597)$