J Netw Syst Manage (2015) 23:1067–1103
DOI 10.1007/s10922-014-9332-6

Two Heuristics for Calculating a Shared Risk Link Group Disjoint Set of Paths of Min-Sum Cost

Teresa Gomes · Miguel Soares · José Craveirinha · Paulo Melo · Luísa Jorge · Vitor Mirones · André Brízido

Received: 9 April 2014 / Revised: 13 September 2014 / Accepted: 22 September 2014 /
Published online: 2 October 2014

Abstract A shared risk link group (SRLG) is a set of links which share a common risk of failure. Routing protocols in Generalized MultiProtocol Label Switching, using distributed SRLG information, can calculate paths avoiding certain SRLGs. For single SRLG failure an end-to-end SRLG-disjoint path pair can be calculated, but to ensure connection in the event of multiple SRLG failures a set with more than two end-to-end SRLG-disjoint paths should be used. Two heuristic, the Conflicting SRLG-Exclusion Min Sum (CoSE-MS) and the Iterative Modified Suurballes's

T. Gomes (✉) · J. Craveirinha
Department of Electrical and Computer Engineering, University of Coimbra, 3030-290 Coimbra, Portugal
e-mail: teresa@deec.uc.pt

J. Craveirinha
e-mail: jcrav@deec.uc.pt

T. Gomes · J. Craveirinha · P. Melo · L. Jorge
INESC Coimbra, 3030-033 Coimbra, Portugal
e-mail: pmelo@fe.uc.pt

L. Jorge
e-mail: ljorge@inescc.pt

M. Soares · V. Mirones · A. Brízido
PT Inovação, 3810-106 Aveiro, Portugal
e-mail: Miguel.Soares.ext@coriant.com

V. Mirones
e-mail: Mirones@ptinovacao.pt

A. Brízido
e-mail: andre-d-brizido@ptinovacao.pt

Present Address:
M. Soares
Coriant Portugal, 2720-093 Amadora, Portugal

Heuristic (IMSH), for calculating node and SRLG-disjoint path pairs, which use the Modified Suurballes's Heuristic, are reviewed and new versions (CoSE-MScd and IMSHd) are proposed, which may improve the number of obtained optimal solutions. Moreover two new heuristics are proposed: kCoSE-MScd and kIMSHd, to calculate a set of k node and SRLG-disjoint paths, seeking to minimize its total cost. To the best of our knowledge these heuristics are a first proposal for seeking a set of k $(k > 2)$ node and SRLG-disjoint paths of minimal additive cost. The performance of the proposed heuristics is evaluated using a real network structure, where SRLGs were randomly defined. The number of solutions found, the percentage of optimal solutions and the relative error of the sub-optimal solutions are presented. Also the CPU time for solving the problem in a path computation element is reported.

Keywords Diverse routing · SRLG-disjoint · Node-disjoint · Min-sum

1 Introduction

Nowadays, due to the very high bandwidth provided by optical networks, the volume of traffic carried in these networks is extremely large. As such, a failure even during a short period of time can leave a very large number of users without service. This can represent a loss of revenue and reputation for the service provider. Hence not only are networks built with automatic recovery schemes but there is also a trend for investing in technologies that may enable the networks recovering from faults before they are perceived by the users.

Restoration is a type of recovery scheme to be used when the affected services can tolerate quality of service (QoS) degradation (such as increased delay or even packet loss) due to the network recovery mechanism. With restoration no backup bandwidth is pre-reserved, and the recovery path (or paths) are only computed and signaled after fault detection. Protection is the preferred recovery solution whenever faults, in certain network elements, should not be perceived by the supported service. In this case an active path (AP), the path that carries traffic under normal conditions, is established and signaled simultaneously with the backup path (BP), which carries traffic when some failure affects the AP.

Recovery can be global, when an end-to-end disjoint BP is calculated; or local if the node closest to the point where the fault occurred, is responsible for the AP recovery [1]. There is also the possibility of dividing a path in segments (that may partially overlap) and ensure locally the protection of each of those segments [2, 3].

A useful concept in network protection is the concept of Shared Risk Link Group (SRLG). An SRLG is a set of links sharing some physical resource (cable, conduit, node, etc.) the failure of which results in the failure of all links of the group [4–6]. Note that a link may be affected by different risks, and as such may belong to

P. Melo
Faculty of Economics, University of Coimbra, 2004-512 Coimbra, Portugal

L. Jorge
Polytechnic Institute of Bragança, 5301-857 Bragança, Portugal

different SRLGs. An SRLG is a general concept that also allows to capture geographically correlated faults, which may result from the links being located in the same seismic or flooding area. The routing protocols in Generalized MultiProtocol Label Switching (GMPLS) networks support distribution of information regarding the SRLG network [6]. The Internet Engineering Task Force (IETF) is working towards a standard for the Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) [7, 8] to support automatic collection of SRLG information [9] for the traffic engineering (TE) link[1] formed by a label switched path (LSP).

A path computation element (PCE) [11] is a computational unit (in a MPLS or GMPLS network) that calculates a path at the request of a path computation client (PCC). A PCE is a network element that determines one or multiple paths in the network domain to which it belongs. To determine a path, a PCE must resort to the traffic engineering database (TED) containing information on the network status. The route calculation can be performed in a centralized or distributed approach [11]. For distributed approaches, in [12] the advantages of pre-reservation mechanisms, when resource status in the network elements differ from the information in the TED, are discussed. A centralized PCE usually has higher processing capabilities, and its response time can be on the order of seconds since the system responds to requests from network management and not directly to changing network conditions. However, if the PCE in question is located "on router"—as it can happen in a distributed model—the calculation power and memory resources available to the PCE are limited, but at the same time it should be able to provide a rapid response. For end-to-end protection in GMPLS networks considering SRLG information the PCE must be capable of calculating SRLG-disjoint path pairs, in a single failure scenario, or a set of k SRLG-disjoint paths in a $k-1$ failures scenario. This shows the importance of developing efficient algorithms for determining SRLG-disjoint paths.

The calculation of a pair of paths disjoint in the arcs (in the nodes) with minimum additive cost, can be used to minimize the cost of resources used in dedicated global protection. This problem, called min-sum, is solved in polynomial time by using the algorithms of Suurballe [13] or Bhandari [14]. When the goal is to share backup bandwidth, the considered problem formulation is usually the min-min problem, where one seeks to determine the minimal cost AP for which a BP can be obtained. This problem is NP-complete [15]. If the total used bandwidth should be minimized the problem is rather difficult to solve, because the amount of backup bandwidth used by a BP depends on the selected AP [16]. In the context of sharing bandwidth methods, the use of network resources by the AP can be considered more important than for the BP. This can lead to min-sum problems with asymmetric weights, wherein the AP cost is considered ω times more important than the cost of BP [17]. This problem is also NP-complete [15]. In [18] an algorithm, designated as $\alpha+1$ protection, is proposed for a partial bandwidth protection scheme, where α is the ratio of the protection bandwidth (of the BP) to the full bandwidth (of active path).

[1] As defined in [10] "A TE link is a "logical" link that has TE properties. The link is logical in a sense that it represents a way to group/map the information about certain physical resources (and their properties) into the information that is used by Constrained SPF for the purpose of path computation, and by GMPLS signaling."

A closely related problem is the min-sum problem in a dual-cost network, that is a network where every edge has two costs with an arbitrary relationship, which is also NP-Complete [19]. In [20] an exact algorithm for solving this problem was proposed, and in [21] a new approach for finding k-disjoint paths with differentiated path cost is presented.

If SRLG information is available, more realistic resilient routing models can be considered where the min-sum and min-min problems are formulated by considering that the paths must be SRLG-disjoint. In this case, the min-sum problem becomes NP-complete [22]. Hence, various heuristics have been proposed for their resolution, some of which are reviewed in Sect. 2. The determination of a set of k SRLG-disjoint paths was considered in [23], where the minimization of the cost of the resulting set was not an objective.

In this work we develop two heuristics for calculating a set of k node and SRLG-disjoint paths, seeking to minimize its total cost. To the best of our knowledge these heuristics are a first proposal for seeking a set of $k\,(k > 2)$ node and SRLG-disjoint paths of minimal additive cost. The performance of the proposed heuristics is evaluated using a real network structure, where SRLGs were randomly defined. The number of solutions found, the percentage of optimal solutions and the relative error of the sub-optimal solutions, are presented. The CPU time for solving the problem in a specific type of PCEs, is also reported. Results will show that the proposed heuristics are effective procedures in terms of discovered solutions and of the relative error of the sub-optimal solutions cost, taking into account the computational limitations of the PCEs.

The major contributions of the paper are the following:

- Development of new versions of two previous heuristics for calculating SRLG-disjoint pairs of minimal additive cost (COSE-MScd and IMSHd) which may improve the number of obtained optimal solutions.
- Proposal of two new heuristics for tackling a difficult combinatorial problem concerning the calculation of a set of $k\,(k > 2)$ node and SRLG-disjoint path pairs of minimal additive cost. To the best of our knowledge these heuristics are the first effective proposal for tackling this problem, which has great potential interest in GMPLS networks.
- Presentation of the ILP formulation of the addressed problem, enabling the evaluation of the optimality of the solutions obtained by the heuristics, in realistic test networks.
- An extensive experimentation study in a real network provided by Portugal Telecom Inovação, enabling the evaluation of the quality of the solutions provided by the two heuristics, by comparison with exact solutions and the running times in a realistic application scenario. These CPU times were obtained considering a real PCE with clear computational limitations, a Desktop using the heuristics and the ILP solution given by a CPLEX solver.

The remaining of the paper is organized as follows. In Sect. 2 a brief review of literature concerning the determination of SRLG-disjoint paths is presented. In Sect. 3 the notation is introduced and the problem of finding a set of k node and SRLG-

disjoint paths of minimal additive cost, is formulated. A review of IMSH and CoSE-MS algorithms (necessary for the comprehension of the developed algorithms) is in Sect. 4. The proposed heuristics are described in Sect. 5 and results using a real network are given in Sect. 6. The conclusions are given in Sect. 7.

2 Related Work

There is a vast literature related to survivable routing problems considering SRLG information. Although an overview of this broad area is out of the scope of this paper we present here a brief review of references concerning the determination of SRLG-disjoint paths, while a more detailed description of works more closely related to the proposed heuristics will be presented in Sect. 5.

A simple approach for solving the min-min SRLG-disjoint path pair problem is the calculation of the shortest path (the AP), followed by the removal of all the links in SRLG conflict with the AP (that is the links that belong to an SRLG in common with the AP) before the calculation of the BP in this pruned network. If no BP can be found, it is said that the algorithm has fallen into a trap. Traps are said to be real [24] if no SRLG-diverse path pair exists due to connectivity issues; however if an SRLG-disjoint path pair exists in the network, but the algorithm can not find it, the algorithm has fallen into an avoidable trap. The number of (avoidable) traps that this type of algorithm falls into can be mitigated by using a k-shortest path enumeration algorithm for generating AP candidates, and then using a similar approach for seeking the BP. This is a form of the iterative two step approach (ITSA) [25].

The trap avoidance (TA) algorithm, proposed in [24], is very effective at avoiding traps. For each new connection request it considers two copies of the network. The first copy is used to calculate the candidate AP with a shortest path algorithm, and the second is used to calculate the corresponding BP. In the second network (which always starts as a copy of the original network) the TA algorithm removes all AP directed links and changes to a large value the cost of the arcs of the reversed links of the AP and also the cost of the all links that share an SRLG with the AP. Because shared path protection is being used, the costs of the remaining links are changed to the bandwidth required (in each of them) to protect the candidate AP. The BP is then calculated in the modified network and a set T, which represents the links of the BP which have an SRLG in common with the links in the AP, is obtained. If T is empty the algorithm ends with the solution AP/BP. If T is not empty, the authors define the most risky active link (of the AP) belonging to the set T and remove it from the first copy of the network. Then algorithm TA begins a new iteration to obtain a min-min SRLG-disjoint path pair. So the AP is calculated in a successively pruned network, and the number of iterations of TA is limited by the number of links in the network.

The Conflicting SRLG-Exclusion (COSE) [26] is also an efficient heuristic for addressing the min-min problem, considering SRLGs. It extends the Conflicting Link Exclusion (CoLE) algorithm [15], replacing the conflicting link set (the set of links to be successively excluded in trying to solve the min-min problem) by the

calculation of the Conflicting SRLG Set (CoSE). In [27, 28] the COSE heuristic was modified to solve the min-sum SRLG-disjoint problem and the resulting heuristic was designated Conflicting SRLG-Exclusion Min-Sum (CoSE-MS). This heuristic is reviewed in detail in Sect. 4.3. The Iterative Modified Suurballe's Heuristic (IMSH) [29] also seeks to solve a min-sum problem consisting of the calculation of an SRLG-disjoint pair of paths, with minimum additive cost, and is reviewed in detail in Sect. 4.2.

In [30] several approaches are proposed for solving the survivable routing problem in optical networks with shared protection, considering SRLGs. They formulate the associated min-sum problem as an Integer Linear Programming (ILP) process, which is not scalable with the network size. Hence the authors propose two heuristics, designated ITSA and maximum likelihood relaxation (MLR). Their simulation results show that the ITSA scheme can achieve the best performance at the expense of more computation time, while MLR can be considered a compromise between computation efficiency and performance.

In [23] the algorithm weighted-SRLG (WSRLG) based on a "*k*-shortest path algorithm with SRLG" is considered, where costs are assigned to the links taking into account the cost of the link and the sum of number of links in the SRLGs the link belongs to (designated as the link SRLG members). The "*k*-shortest path algorithm with SRLG" first calculates the shortest path in the network; then prunes the links of that path and the links belonging to the SRLGs in the path, and calculates a shortest path in the resulting network; this process is repeated until no additional paths can be calculated. The WSRLG algorithm makes a binary search of the weights used to define the cost of the links, depending on the size of the set of SRLG-disjoint paths most recently obtained by the "*k*-shortest path algorithm with SRLG". The algorithm ends when the binary search is considered to have converged. Then, among the set of obtained paths, it selects the one the size of which is closer to the target size, and among those of equal size the one with minimal additive cost.

The authors in [31] consider SRLGs and Shared-Risk Node Groups (SRNGs), and define shared risk resource group (SRRG) failures. They propose graph transformation techniques which converts the SRRG-disjoint path pair problem into a node-disjoint path pair problem, for certain restricted SRRGs, and hence provide a polynomial time solution for specific cases.

In [32] it is considered that once an SRLG failure event occurs, its associated links fail with some probabilities, thus resulting in the definition of a Probabilistic SRLG (PSRLG). This framework, representing probabilistic correlated failures, is considered by the authors to be more adequate for coping with erroneous SRLG data, that may occur due to traffic engineering and recovery mechanism. Additionally, mathematical formulations and heuristics for the problem of diverse routing with minimum joint failure probability were developed [32]. Diaz et al. [33] remark that the approach proposed in [32] focuses on risk minimization and ignores traffic engineering issues. A solution designated as the joint path pair load balancing (JPP-LB) scheme is hence proposed [33], which seeks to balance risk minimization in a multi-failure scenario and traffic engineering constraints.

In [34, 35], a two-step approach is used to solve the optical network diverse provisioning problem. In the first step, the diverse routing problem is formulated using ILP to find optimal SRLG-diverse routes with the minimum objective value (either cost or distance). Additionally, the ILP formulation was extended in order to address the multiple-objective optimization problem of obtaining maximally SRLG-diverse routes, when no SRLG-disjoint solution exists. The second step consists of a dense wavelength division multiplexing (DWDM) system selection, regenerator placement and wavelength assignment, without changing the cost determined in the previous step.

The design of disaster-resilient optical datacenter networks is addressed in [36], where the authors use the concept of Shared Risk Group (SRG) to define potential disaster zones. To ensure disaster protection in optical networks, active and backup light paths as well as multiple locations of content/services must be SRG-disjoint. In [36] an integrated ILP formulation to design datacenter networks while ensuring single disaster survivability is proposed, which solves simultaneously the problem of content placement, resilient routing and content disaster protection. The authors also propose ILP relaxations and heuristics to solve problems for large networks.

A new ILP formulation to solve the resilient grid/cloud dimensioning problem, comprising both network and server resources, for large-scale decentralized distributed systems is proposed in [37]. The concept of SRLG is used to represent the survivability requirement, where the links model either optical network links (network failures), or represent the connection to the data center (server failures). They consider both failure-dependent (FD) rerouting, where backup routes (and server locations) may be chosen differently for different failure cases, and failure-independent (FID) routing with a single BP and destination for all failure cases. They conclude that, in the problem they considered, FD does not bring significant benefits compared to FID.

3 Notation and Problem Formulation

3.1 Notation

The heuristics in Sect. 5 use the following notation. Let the graph $G = (V, A)$ be defined by a set of nodes V, $V = \{v_1, \ldots, v_n\}$, and a set of arcs A, $A = \{a_1, \ldots, a_m\}$.

An arc connects two vertexes in a given order, and is an ordered pair of elements belonging to V. If $v_i, v_j \in V$, with $v_i \neq v_j$ and $a = (v_i, v_j) \in A$, it is said that the v_i is the tail (or source) of the arc and v_j is its head (or destination). Arc (v_i, v_j) is said to be emergent from node v_i and incident on node v_j. Arcs (v_i, v_j) and (v_j, v_i) are symmetrical arcs.

The cost of using an arc $(v_i, v_j) \in A$ in a path is given by $l(v_i, v_j)$, and is assumed to be strictly positive.

A path is a continuous sequence of nodes (all different) from one node source, s, to a destination node t, $(s, t \in V)$, and is represented by $p = \langle s \equiv v_1, v_2, \ldots, v_u \equiv t \rangle$, where $(v_i, v_{i+1}) \in A, \forall i \in \{1, \ldots, u-1\}$, u being the number of nodes in the path. Let V_p be the set of nodes in the path p and A_p be the set of arcs that form the path,

$A_p = \cup_{\forall i \in \{1,\ldots,u-1\}}(v_i, v_{i+1})$. A segment is a continuous sequence of arcs that are part of a path. The set of arcs symmetrical of the arcs in A_p is $\bar{A}_p$.

The additive cost of a path p is the sum of the costs of the arcs constituting the path, $c_p = \sum_{(v_i,v_j)\in A_p} l(v_i, v_j)$. If a path between a given pair of nodes does not exist, is represented by the empty set $(\emptyset)$, and its cost is infinite.

Given a node pair (s,t), a pair of paths from s to t is represented by (p,q). The paths are node disjoint if and only if $V_p \cap V_q = \{s,t\}$.

A set of k paths with the same node source s and destination t is represented by S, where $k = |S|$. The paths in the set S (from s to t), are mutually node disjoint, if and only if: $\cap_{p\in S} V_p = \{s,t\}$.

The additive cost of a pair of paths (p,q) is given by the sum of the cost of the paths forming the pair, $c_{(p,q)} = c_p + c_q$. If $(p,q) = (\emptyset,\emptyset)$, the cost of the pair of paths is infinite $(c_{(\emptyset,\emptyset)} = \infty)$. The cost of a set of paths S is given by the sum of the cost of the paths in this set, $c_S = \sum_{p\in S} c_p$.

Let Y, with $Y = \{y_1, y_2, \ldots, y_r\}$ designate the set of failure risks that may affect the arcs of the network, where r is the number of risks in the network. The set of arcs of the network that become unavailable when the failure associated with risk y_i occurs is the SRLG g_i, $i = 1,\ldots,r$. Let R' be the set of all SRLGs of the network. $R' = \{g_1, g_2, \ldots g_r\}$ where r is the number of SRLGs in the network.

Let $R(v_i, v_j)$ or $R(a)$ with $a = (v_i, v_j) \in A$ be the set of SRLGs which contain arc $a = (v_i, v_j)$ $(R(a) = \{g_u : a \in g_u\})$. From the above definitions: $R' = \cup_{a\in A} R(a)$. The set of SRLGs affecting a path p is designated by R_p and is given by $R_p = \cup_{a\in A_p} R(a)$. A path pair (p,q) is SRLG-disjoint if $R_p \cap R_q = \emptyset$. The arcs that are in SRLG conflict with the arcs along a path p are the arcs in $(\cup_{g\in R_p} g)\backslash(A_p \cup \bar{A}_p)$.

Let $\mathscr{P}_{st}$ represent the set of all paths from s to t in the network.

The set of k paths, from s to t, which are node and SRLG-disjoint of minimal additive cost is designated by S^*.

3.2 Problem Formulation

The problem of calculating the set of k paths, from s to t, which are node and SRLG-disjoint of minimal additive can be stated as follows:

$$S^* = \arg\min_{S\subset\mathscr{P}_{st}} c_S \tag{1}$$

$$\text{such that: } \cap_{p\in S^*} V_p = \{s,t\} \tag{2}$$

$$\cap_{p\in S^*} R_p = \emptyset \tag{3}$$

$$|S^*| = k \tag{4}$$

The ILP formulation for obtaining S^* is given here, because the exact results obtained using this formulation will be used to evaluate the performance of the heuristics. The formulation is inspired on the one by [22]. The formulation requires some additional notation:

- $\delta(i)^+$: set of arcs in A emergent from node $v_i \in V$.
- $\delta(i)^-$: set of arcs in A incident on node $v_i \in V$.
- $h_{g,(i,j)}$, with $g \in R'$ and $(v_i, v_j) \in A$, indicates if SRLG g contains arc (v_i, v_j)

$$h_{g,(i,j)} = \begin{cases} 1 & \text{if } g \in R(v_i, v_j), \\ 0 & \text{otherwise ;} \end{cases} \tag{5}$$

- $x_{(i,j),u}$ is the binary decision variable of arc $(v_i, v_j) \in A$ associated with path p_u $(u = 1, 2, \ldots, k)$, where,

$$x_{(i,j),u} = \begin{cases} 1 & \text{if arc } (v_i, v_j) \in A_{p_u}, \\ 0 & \text{otherwise ;} \end{cases} \tag{6}$$

- $z_{g,u}$ is the binary decision variable of the SRLG which affects path p_u $(u = 1, 2, \ldots, k)$, where,

$$z_{g,u} = \begin{cases} 1 & \text{if } g \text{ is associated with path } p_u, \text{ that is, if } g \in R_{p_u}, \\ 0 & \text{otherwise ;} \end{cases} \tag{7}$$

The problem of obtaining a set of node and SRLG-disjoint solution paths of minimal cost, from node s to t can be formulated as follows.

$$\min \sum_{(v_i,v_j)\in A} l(v_i, v_j) \left(\sum_{u=1}^{k} x_{(i,j),u} \right) \tag{8}$$

$$\text{such that: } \sum_{(v_i,v_j)\in\delta(i)^+} x_{(i,j),u} - \sum_{(v_j,v_i)\in\delta(i)^-} x_{(j,i),u} = \begin{cases} 1 & : \quad v_i = s, \\ -1 & : \quad v_i = t, \\ 0 & : \quad v_i \in V \backslash \{s, t\} \end{cases} \tag{9}$$

$$v_i \in V,\ u = 1, 2, \ldots, k$$

$$\sum_{(v_i,v_j)\in A} h_{g,(i,j)} x_{(i,j),u} \le |A| z_{g,u} \quad g \in R',\ u = 1, 2, \ldots, k \tag{10}$$

$$\sum_{u=1}^{k} z_{g,u} \le 1, \quad g \in R' \tag{11}$$

$$\sum_{u=1}^{k} \sum_{(v_i,v_j)\in\delta(i)^+} x_{(i,j),u} \le 1, \quad v_i \in V \backslash \{s\}, \tag{12}$$

x and z are the binary decision variables.

- Constraint (9) ensures that arcs (v_i, v_i) selected by $x_{(i,j),u}$, are a path p_u $(u = 1, 2, \ldots, k)$ from s to t.
- Constraint (10) implies that if g affects path p_u, then any arc belonging to g can be in p_u; otherwise no edge in g can be in p_u. The coefficient $|A|$ is used, because p_u can contain several arcs associated with a given SRLG.
- Constraint (11) ensures that no SRLG affects more than one path in a set S^* with k paths.
- Constraint (12) ensures the paths are node disjoint.

4 Review of IMSH and CoSE-MS

Since modified versions of heuristics IMSH and CoSE-MS were used as a basis for developing of our heuristics, they are reviewed in this section. Note that both the Modified Suurballe's heuristic and the modified Bhandari's heuristic (MBH), are used in CoSE-MS. Both auxiliary heuristics are also over-viewed in the next subsection.

4.1 Auxiliary Heuristics

In [29] a modification of Suurballe's algorithm [13] is proposed, designated as Modified Suurballe's Heuristic (MSH), which can be applied to the u-th shortest path for obtaining a pair of edge and SRLG-disjoint paths. Here it is revisited for obtaining node and SRLG-disjoint path pairs.

In MSH a new modified graph, $G' = (A', V')$, is derived from to G where p_u was calculated. Because node and SRLG-disjoint paths are sought, the first step is the replacement in G' of all the intermediate nodes of p_u by an arc. So for all $v_i \in V_{p_u} \backslash \{s, t\}$, v_i is replaced by arc (v_i', v_i'') with null cost, and all the arcs in $\delta(i)^+$ will now emerge from v_i'' and all the arcs in $\delta(i)^-$ will now be incident on v_i'. This corresponds to using one of the vertex-splitting methods described by [14].

In G' the arcs in A_{p_u} and $\bar{A}_{p_u} \cap A'$ are removed before adding the arcs $\bar{A}_{p_u}$ with null cost; then the cost of the arcs in the graph which are in SRLG conflict with the arcs along the path p_u is increased by M (sum of the costs of all the arcs in the network). The shortest path in this network, q_u', is calculated and the divided nodes from p_u (if present in q_u') are collapsed into the original node. As in Suurballe's algorithm every directed arc in q_u' the reversal of which appears in p_u is designated as an interlacing arc. These interlacing arcs must be removed from paths p_u and q_u' to get a pair of least cost node-disjoint paths. The obtained path pair (p_u', p_u'') is only considered an admissible solution if p_u' and p_u'' are SRLG-disjoint. An illustrative example of the MSH behavior can be found in Appendix 1.

The MBH proposed by [27] can only be applied to the shortest path in G, because it uses negative costs. The version of the MBH used in this work seeks to obtain node-disjoint path pairs of min-sum cost. Hence, in the G' graph (identical to G) where p_1 was calculated, MBH starts by splitting the nodes as described for the

MSH. Then, as in MSH the arcs in A_{p_1} and $\bar{A}_{p_1} \cap A'$ are removed, the arcs in $\bar{A}_{p_u}$ (directed arcs from t to s in p_1) are added, but each with the symmetrical of the cost of the corresponding symmetrical arc in p_1. Then the shortest path in this new network, q'_1, is calculated, using the modified Dijkstra's algorithm [14]. Finally one must remove the interlacing arcs on paths p_1 and q'_1 to get a pair of least cost node-disjoint paths. Although, as in the MSH the calculated path pair may not be SRLG-disjoint, MBH tends to find solutions with lower cost than MSH.

4.2 Review of IMSH

The Iterative Modified Suurballe's Heuristic sequentially generates v shortest path using Yen's algorithm [38]; then for each obtained p_u (u-th shortest path, $u = 1, \ldots, v$) it uses the MSH to calculate a pair of SRLG-disjoint paths based on each p_u, and keeps a record of the path pair with current lowest additive cost. The algorithm ends after generating v shortest paths or earlier if the recorded SRLG-disjoint path pair (current best solution) was obtained, (p, q) is such that $c[(p, q)] \leq 2 \times p_u$. Although in [29] a proof is presented that the condition $c[(p, q)] \leq 2 \times p_u$, ensures the optimality of (p, q), the example in Appendix 3, shows that this condition may not hold if the SRLG are randomly generated. Hence in our tests the number of generated seed paths is defined by the maximum number of allowed iterations $(i_{\max})$ or the number of existing paths in the network.

4.3 Review of CoSE-MS

The CoSE-MS algorithm operates by solving problems which are represented by $P(I, E, H)$, where I is the inclusion set of SRLGs, E the exclusion set of SRLGs and H the union of all the exclusion sets of the problems that originated the current problem P. Together E and H represent the set of SRLG that have to be excluded from the network before the calculation of candidate seed path: the first contains the most recent SRLG signaled for exclusion and the second all the previously excluded SRLGs. If the candidate seed path does not allow to obtain an SRLG-disjoint path pair, new problems are generated, but the SRLGs in set I can not be excluded in the new problems to be generated. The algorithm successively divides the SRLGs into disjoint subsets: the exclusion SRLGs $(E \cup H)$ and the inclusion SRLGs (I). The problems are stored in a stack (S_P), and CoSE-MS will try to solve problems until the stack is empty, or until a certain number $(i_{\max})$ of problems have been solved.

The initial problem will have I, E and H equal to $\emptyset$, and is pushed into a previously empty stack (S_P) of unsolved problems. In each iteration the heuristic gets (and removes) the problem from the top of stack S_P; let that problem be the current problem $P(I_c, E_c, H_c)$. Its resolution is described next.

The seed path p_c of the current problem is calculated in graph G_c, corresponding to the original network graph G where all arcs affected by SRLGs in the set $E \cup H$ have been removed. If p_c can not be found the problem resolution ends with no solution.

If p_c is the shortest path in the original network graph G, that is if $(I_c, E_c, H_c) = (\emptyset, \emptyset, \emptyset)$ then the MBH is used; otherwise the MSH is used. In both cases, MBH or MSH, will modify a copy G' of the original network G.

If the seed p_c results (using MBH or MSH) in the path pair (p'_c, p''_c), and that path pair is SRLG-disjoint, then a solution was found—the algorithms will store the best solution found so far.

If no path pair can be obtained using p_c as seed path, or the resulting path pair (p'_c, p''_c) is not SRLG-disjoint, the conflicting SRLG set, T_c is calculated, and new problems are generated. The conflicting SRLG set T_c is the subset of $R_{p_c} \backslash I_c$ such that no path exists from s to t in the network graph G after the removal of the arcs in the SRLGs in T_c. The set T_c can be calculated using the algorithm "Finding a conflicting SRLG set for a given AP p_c from node s to node t" in [26] (also in Appendix of [27, 28]). Let $T_c = \{g_1^c, g_2^c, \ldots g_{|T_c|}^c\}$, then the following new problems are generated in CoSE-MS: $P(\emptyset, \{g_1^c\}, E_c \cup H_c)$, $P(\{g_1^c\}, \{g_2^c\}, E_c \cup H_c)$, …, $P(\{g_1^c, g_2^c, \ldots, g_{|T_c|-1}^c\}, \{g_{|T_c|}^c\}, E_c \cup H_c)$, and pushed into the stack S_P of problems.

When the solution of problem P_c is a node-disjoint, but not SRLG-disjoint path pair (hence a solution not admissible), each of the new problems derived from T_c has one more SRLG to be excluded than P_c, and the convergence of the heuristic is ensured.

5 Proposed Heuristics

5.1 New Version of MSH and IMSH

When the SRLGs are not strictly local (see Appendix 3), in MSH a seed path p_c may result in a path pair (p'_c, p''_c) which is not SRLG-disjoint, even though the path q'_c obtained in G' had no arc of cost greater than M. So although p_c and q'_c may be SRLG-disjoint, after removing the interlacing arcs the resulting path pair may have one or more SRLGs in common.

Also note that if in the network G' (after the transformations required by the MSH) there are alternative shortest paths with cost lower than M, there are two possible scenarios for each candidate shortest path: (a) the path contains reversed arcs of p_c; (b) the path does not contains reversed arcs of p_c. In case (a) the resulting path pair will have the lowest cost (for the used p_c) but the resulting path pair may not be SRLG-disjoint; in case (b) the cost of the path pair will be higher than in case (a) but the resulting path pair will be SRLG-disjoint.

If the cost of arcs in the network are strictly positive, and that is usually the case in real networks, then we propose the following variant of MSH, designated by MSHd, which instead of setting a null cost to arcs of $\bar{A}_{p_c}$ will set the cost of these arcs equal to $-\Delta$ (where $\Delta = \min_{(v_i,v_j)\in A} l(v_i, v_j)/(2|V|)$, is a very small number), given preference to the path in case a). If the resulting path pair is SRLG-disjoint, MSHd ends, otherwise if there was interlacing removal, the edges with cost $-\Delta$ are changed to $+\Delta$ in order to obtain a solution of type b), in case it exists.

The version of IMSH which uses MSHd instead of MSH, is designated by IMSHd.

5.2 New Version of CoSE-MS

For some node pairs, CoSE-MS generates a large number of problems. This suggested that it could be improved, by changing the calculation of the set of SRLG in conflict, T_c. In fact when the resulting path pair (p'_c, p''_c) is node-disjoint but not SRLG-disjoint, the conflicting SRLG set should depend on $R_{p'_c} \cap R_{p''_c}$ and not only on R_{p_c}. This resulted in the following new approach to the calculation of T_c:

- If the problem is $(\emptyset, \emptyset, \emptyset)$ or no node disjoint path pair could be obtained, the conflicting SRLG set is calculated as described in Sect. 4.3.
- Otherwise, if the resulting path pair (p'_c, p''_c) is node-disjoint but not SRLG-disjoint, T_c will be given by $(R_{p_c} \cap X) \backslash I_c$, where $X = R_{p'_c} \cap R_{p''_c}$.

The version of CoSE-MS with this new procedure for calculating T_c and with MSHd replacing MSH, is designated by CoSE-MScd.

In [28] the MSH and the MBH returned $(\emptyset, \emptyset)$ whenever the resulting path pair was not SRLG-disjoint. In the present resolution procedure, when the resulting path pair is node-disjoint but not SRLG-disjoint, the path pair is returned by MSHd and MBH, so that X can be calculated.

5.3 Two New Heuristics, kIMSHd and kCoSE-MScd

In this subsection two heuristics for calculating a set of k node and SRLG-disjoint paths, of minimal additive cost, will be presented.

These heuristics, designated kIMSHd and kCoSE-MScd, require three main steps:

1. Calculation of a set of node and SRLG-disjoint path pairs, which will be used as the seed set.
2. For each element of the seed path calculated in the previous step, a set of k node and SRLG-disjoint paths is calculated; if that dimension k is not attained, the set (or sets) of greater dimension are stored.
3. The set of minimal cost is selected among all those sets of size k (or among the sets of largest dimension, less than k, that could be found).

Obtaining a set of k node and SRLG-disjoint may not be possible, because either it does not exist, or because the heuristics were unable to find such a set. In this case the heuristics will return the best solution they could find, even if its size was not k. The application that invoked the heuristics should then decide whether to reject or accept that solution.

The heuristics kIMSHd and kCoSE-MScd differ in step 1. In heuristics kIMSHd and kCoSE-MScd the seed set is obtained storing all the node and SRLG-disjoint

path pairs that are generated during the execution of heuristics IMSHd and CoSE-MScd, respectively.

Data: Digraph $G = (V, A)$; source node s; target node t; arc cost $l(v_i, v_j)$; $R(v_i, v_j)$, the SRLG associated with each arc $(v_i, v_j) \in A$; seed set of node and SRLG-disjoint paths S_c
Result: Set of node and SRLG-disjoint paths S, obtained using the seed set S_c, with $|S| = |S_c| + 1$ or $S = \emptyset$ if it was not possible to enlarge S_c.
1 $G' \leftarrow G$ a copy of G is used, with ($G' = (V' = V, A' = A)$ // Network transformation
2 The vertex-splitting of all intermediates nodes of the paths in S_c is done (as described for the MSH in section 4.1)
3 $A_{S_c} = \cup_{p \in S_c} A_p$
4 $\bar{A}_{S_c} = \cup_{p \in S_c} \bar{A}_p$
5 In G' the arcs in A_{S_c} and $\bar{A}_{S_c} \cap A'$ are removed before adding the arcs $\bar{A}_{S_c}$ with null cost.
6 The cost of the arcs in the graph that are in SRLG conflict with the arcs in S_c is increased by M (sum of the costs of all the arcs in the network)
// Shortest Path is the transformed network
7 $q' \leftarrow$shortest path from s to t in G' // Using Dijkstra's algorithm [39]
8 **if** q' *exists* **then**
// Remove possible existing interlacing arcs
9 Create a graph G_I induced by the arcs in q' and S_c
10 Remove from G_I all pairs of symmetrical arcs
11 In this modified graph, G_I, calculate the set S of $|S_c| + 1$ node disjoint paths
12 **if** $\cap_{p \in S} R_p \neq \emptyset$ // If the paths in S are not SRLG-disjoint
13 **then**
14 $S \leftarrow \emptyset$ // No solution
15 **end**
16 **end**
17 **else**
18 $S \leftarrow \emptyset$ // No solution
19 **end**

Heuristic kMSH: Algorithm of kMSH, which given a set S_c of u seed paths, node and SRLG-disjoint, returns a set S of $u + 1$ node and SRLG-disjoint paths, or an empty set if it was unable to enlarge S_c

Step 2 corresponds to an extension of of MSH, and the corresponding heuristic is kMSH. The heuristic kMSH corresponds to the application of MSH to a set with v paths, mutually node and SRLG-disjoint, seeking to obtain a set of $v + 1$ paths, mutually node-disjoint and possibly SRLG-disjoint, while minimizing its total cost (similarly to the algorithm in [14] for obtaining a set of k node disjoint paths of minimal total cost).

In line 6 of the algorithm of kMSH, the arcs in SRLG conflict with the arcs in S_c are $(\cap_{g \in R_{S_c}} g) \backslash (\bar{A}_{S_c} \cap A)$, with $A_{S_c} = \cup_{p \in S_c} A_p$ and $R_{S_c} = \cup_{p \in S_c} R_p$.

5.3.1 kIMSHd Heuristic

The kIMSHd heuristic starts by generating and storing in a stack (P in the algorithm of kIMSHd) all the path pairs, which are node and SRLG-disjoint, generated in iterations $i = 1, \ldots, i_{\max}$ of IMHS (see lines 5-15 of kIMSHd). Then each of the path pairs stored in that stack is used as the seed set in kMSH; this heuristic must be invoked $k - 2$ times, using as input the set that resulted from the previous call of

kMSH; the process is interrupted (for the current seed set) if the outcome of kMSH is an empty set—see the cycle in lines 20–23 of kIMSHd. Finally kIMSH returns the set of minimal cost among the largest obtained sets (of size less than or equal to k).

Data: Digraph $G = (V, A)$; source node s; target node t; arc cost $l(v_i, v_j)$; $R(v_i, v_j)$, the SRLG associated with each arc $(v_i, v_j) \in A$; k, $k > 2$ size of the set; maximal number of iterations $i_{\max}$.
Result: Set of node and SRLG-disjoint paths S, of minimal cost

1 $i \leftarrow 0$
2 $\Delta = (\min_{(i,j)\in A} l(i,j))/(2|V|)$ // Assumes $\min_{(i,j)\in A} l(i,j) > 0$
3 $P \leftarrow \emptyset$ // Stack of node and SRLG-disjoint path pairs
4 $S \leftarrow \emptyset$ // Best set of node and SRLG-disjoint paths
// Storing all node and SRLG-disjoint path pairs generated in IMSHd
5 **while** $i \le i_{\max}$ **do**
6 $i \leftarrow i + 1$ // i-th seed path
7 $(p_i', p_i'') \leftarrow (\emptyset, \emptyset)$ // Initially there is no solution
8 $p_i \leftarrow$MPS(G,s,t,l) // i-th shortest path using MPS
9 **if** p_i' *exists* **then**
10 $(p_i', p_i'') \leftarrow$MSHd(G,s,t,l,R,p_i,Δ) // Possibly a node and SRLG-disjoint pair
11 **end**
12 **if** $(p_i', p_i'') \neq (\emptyset,\emptyset) \wedge (R_{p_i'} \cap R_{p_i''}) = \emptyset$ **then**
13 push$(P, (p_i', p_i''))$ // Stores node and SRLG-disjoint path pair
14 **end**
15 **end**
// Using the sets in P as seed paths, when $i = 2$
16 **while** $\neg$ empty(P) **do**
17 $i = 2$ // Initial size of the set
18 $S_i \leftarrow$ top(P) // Gets the path pair at the top of P
19 pop(P) // Removes the element at the top of P
20 **while** $i < k \wedge S_i \neq \emptyset$ **do**
21 $S_{i+1} =$kMSH(G, s, t, l, R, S_i) // Tries to add a new path
22 $i \leftarrow i + 1$
23 **end**
24 **if** $S_i = \emptyset$ **then**
25 $i \leftarrow i - 1$ // S_i is the largest set from cycle in lines 20--23
26 **end**
// Verifies the need to update the best set
27 **if** $\{i = k \wedge [|S| < k \vee (|S| = k \wedge c_{S_i} < c_S)]\} \vee \{i < k \wedge [(|S| = |S_i| \wedge c_{S_i} < c_S) \vee |S| < |S_i|]\}$ **then**
28 $S \leftarrow S_i$ // Updates the best set
29 **end**
30 **end**

Heuristic kIMSHd: Seeks to obtain a set of node and SRLG-disjoint paths of minimal additive cost, using as seed paths the node and SRLG-disjoint path pairs that would be generated in IMSHd

In [40] the authors state that the use of the theoretical worst-case complexity of QoS routing algorithms should be considered with care, because this is not the best indicator for the execution times in most practical problems. In [41] the authors reinforce the same view, namely regarding the use of algorithm MPS in multiple criteria shortest path models. MPS sorts the edges according to their reduced cost; this results that each time a k-th shortest path is selected, the generation of each new

candidate path requires no network modification, unlike in Yen's algorithm. Hence, although Yen's algorithm has lower worst case complexity than MPS [38, 42], we preferred to use MPS [43, 44] because extensive experimental results show that this algorithm is more efficient than Yen's in practice [44, 45]. In the pseudo-code of the kIMSHd heuristic, MPS represents the k-shortest path enumeration algorithm proposed in [43], in its loopless version.

5.3.2 kCoSE-MScd Heuristic

The kCoSE-MScd heuristic is similar to kIMSHd, with the difference that the set of path pairs in stack P corresponds to the node and SRLG-disjoint path pair obtained using CoSE-MScd. This is represented here by the auxiliary heuristic AllPairs described in Appendix 2. The remaining steps of kCoSE-MScd (from line 4 until the end of kCoSE-MScd) are identical to the block of lines from 16 until the end of kIMSHd.

Data: Digraph $G = (V, A)$; source node s; target node t; arc cost $l(v_i, v_j)$; $R(v_i, v_j)$, the SRLG associated with each arc $(v_i, v_j) \in A$; k, $k > 2$ size of the set; maximal number of iterations $i_{\max}$.
Result: Set of node and SRLG-disjoint paths S, of minimal cost

```
1  S ← ∅ // Set of node and SRLG-disjoint paths
2  Δ = (min_{(i,j)∈A} l(i,j))/(2|V|)                    // Assumes min_{(i,j)∈A} l(i,j) > 0
3  P ← AllPairs(G, s, t, l, R, Δ, i_max)        // Stack of seed path pairs using CoSE-MScd
                                  // Using the sets in P as seed paths, when i = 2
4  while ¬ empty(P) do
5      i = 2
6      S_i ← top(P)
7      pop(P)
8      while i < k ∧ S_i ≠ ∅ do
9          S_{i+1} = kMSH(G, s, t, l, R, S_i)              // Tries to add a new path
10         i ← i + 1
11     end
12     if S_i = ∅ then
13         i ← i − 1        // S_i is the largest set from cycle in lines 8-11
14     end
       // Verifies the need to update the best set
15     if {i = k ∧ [|S| < k ∨ (|S| = k ∧ c_{S_i} < c_S)]} ∨ {i < k ∧ [(|S| = |S_i| ∧ c_{S_i} < c_S) ∨ |S| < |S_i|]} then
16         S ← S_i                                          // Updates the best set
17     end
18 end
```

Heuristic kCoSE-MScd: Seeks to obtain a set of node and SRLG-disjoint paths of minimal additive cost, using as seed paths the node and SRLG-disjoint path pairs generated in CoSE-MS

6 Experimental Results

Here the results obtained in terms of the quality of the solutions and the running times of the heuristics, applied to a real network with randomly generated SRLGs,

are presented. To the best of our knowledge, no other algorithm seeking to minimize the total cost of a set of k node and SRLG-disjoint paths ($k > 2$), was previously proposed. Nevertheless in [23] the authors claim that the Weighted-SRLG path selection algorithm (WSRLG) can obtain "cost-effective disjoint paths", and this is the reason why results will also be presented for WRLG, with $\varepsilon = 10^{-15}$ (error for the binary search of the weight factor for the SRLG, and the smallest possible value due to floating point number representation).

Firstly the test conditions for analyzing the quality of the solutions and for determining the execution times are described. Secondly the results for assessing the quality of the solutions obtained are presented, followed by the execution times of the proposed heuristics.

6.1 Test Conditions and Performance Measures

The test network corresponds to the largest bi-connected component of an SDH network with 231 nodes and 471 edges (each arc will be represented by two symmetrical arcs), provided by Portugal Telecom Inovação. In order to study the quality of the solutions obtained by each of the heuristics, up to 235 SRLGs where considered. Given that the average node degree of the network was 4, it was decided that no SRLG should have more than 4 edges. Like in [23, 46, 47] the SRLGs were also randomly generated. Also note that even if the SRLGs were strictly local (all links of any SRLG share an end-point) the problem of calculating $k\,(k \geq 2)$ SRLG-disjoint paths is in general still NP-complete [48] (with some exceptions). Each edge was randomly associated with between 0 and 4 SRLGs, using a uniform distribution. The SRLGs were randomly built as each edge was associated with the previously calculated number of SRLG identifiers (randomly selected among the SRLG still with less than 4 edges). Ten different sets of random seeds were used, resulting in ten different $R(a), a \in A$, with $|R'| \in [231, 235]$.

In the results presented, the maximal number of iterations considered (for kIMSHd) or maximum number of auxiliary problems solved (in kCoSE-MScd) were $i_{max} = 5, 10, 20, 50, 100, 200, 500, 1{,}000$. The number of paths in each set was $k = 3, 4$. For $k = 5$ only 2.2 % of the node pairs have a node and SRLG-disjoint solution, so this value k was not considered.

In order to study the quality of the solutions obtained by each heuristic using each of the 10 instances of the network, we tried to obtain a solution (set of $k = 2, 3, 4$ node and SRLG-disjoint solutions with minimal additive cost) for all source destination pairs of each image, considering the maximal number allowed problems or iterations. Then the cost of the obtained path pair $(k = 2)$ or set $(k = 3, 4)$ was compared to the cost of the optimal solution obtained by a the resolution of the ILP problem in Sect. 3.2 using CPLEX (version 12.3). With this information it was possible to obtain the average number of solutions found by the heuristics, the average number of optimal solutions, and the relative error of the sub-optimal solutions.

If the heuristics return a set of size less than the desired value k, this is considered as not having solved the problem (and not having found any solution). So, for a

given value of k, only a set of size k is considered an admissible solution, and only those solutions (optimal or sub-optimal) are considered in the statistics presented in the figures of this section. There is no way to calculate an upper bound on $i_{\max}$ (or CPU time) using any of the heuristics, that would allows us to state whether there is no solution for a given problem, except in very specific cases (like in cases where there are less than k node-disjoint paths between the considered end nodes).

The CPU times were measured in two different platforms: a PCE, model UNICOM-V5, G2_LE CPU (PowerPC compatible core) with 330 MHz core clock, 128 MB of RAM; and a Desktop with a Intel® Core*TM* i7 870 CPU, with a 2.93 GHz core clock, 3.6 GB of RAM. A dynamic library in pq2 (for PowerPC microprocessors) was created with the developed heuristics, and linked with the test programs that were to run in the PCE. The CPU times in the Desktop were obtained for CPLEX and the heuristics to evaluate the trade-off between CPU time and accuracy of the heuristics.

For the CPU times in the PCE, 1,000 different end nodes were randomly selected in each network and the CPU time was registered for each value of $i_{\max}$. Regarding the CPU times in the Desktop, 5,000 different end nodes were randomly selected in each network, and the CPU time was registered for each value of $i_{\max}$. The amplitude bars in the figures in the next subsection represent 95 % confidence intervals, using the average values obtained for the 10 networks.

6.2 Quality of the Solutions

To evaluate the quality of the solutions obtained, some results are presented. Note that algorithm WSRLG, using $\varepsilon = 10^{-15}$ will only perform a maximum of 50 iterations. However, in the figures, the results will be presented for all considered values of $i_{\max}$.

In Figs. 1 and 2, although IMSHd leads in average to slightly more solutions than IMSH, taking into account the strong overlapping of the confidence intervals, in practice there are no significant differences between the two heuristics in this respect. A similar statement can be made when comparing CoSE-MScd with CoSE-MS.

Considering $i_{\max} = 50$, the number of optimal solutions found by IMSH and IMSHd is over 95 %, while it is between 80 and 85 % for CoSE-MS and CoSE-MScd and it is close to 65 % for WSRLG. IMSH and IMSHd keep improving the number of obtained solutions and of optimal solutions with the increase in the maximum number of allowed iterations, while CoSE-MS and CoSE-MScd seem to stagnate their performance after $i_{\max} = 100$.

The major conclusion from these results (Figs. 1, 2) is that WSRLG ($k = 2$) has the lowest performance in terms of obtained solutions and optimal solutions by comparison with the other heuristics.

For each node pair (s, t), for which a sub-optimal solution S with cost c_S was obtained, the relative error of the pair, *re*, was calculated as follows:

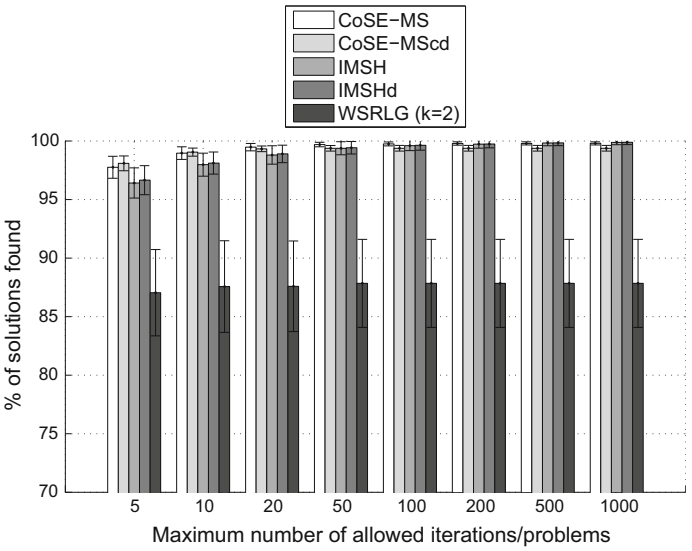

Fig. 1 Average number (%) of solutions found for $k = 2$ by CoSE-MS, CoSE-MScd, IMSH, IMSHd, and WSRLG for $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1{,}000$

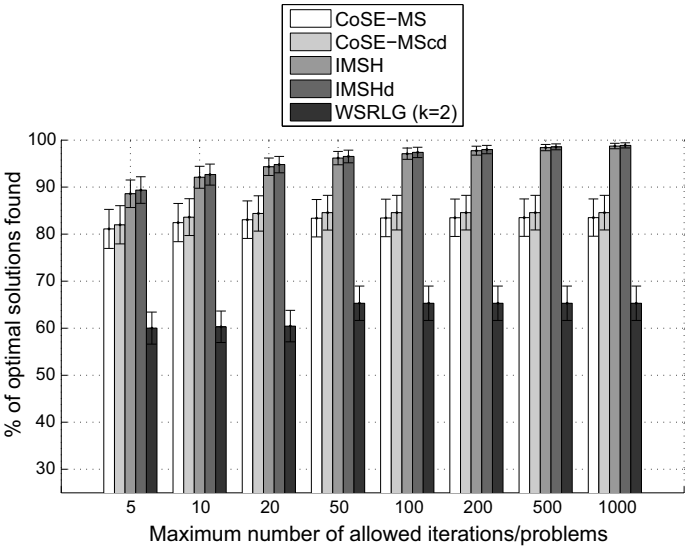

Fig. 2 Average number (%) of optimal solutions found for $k = 2$ by CoSE-MS, CoSE-MScd, IMSH, IMSHd, and WSRLG for $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1{,}000$

$$re = \frac{c_S - c_{S^*}}{c_{S^*}} \tag{13}$$

where S^* represents the node and SRLG-disjoint set of minimal additive cost given by c_{S^*}. In each network, the relative errors (re) for every node pair with a sub-

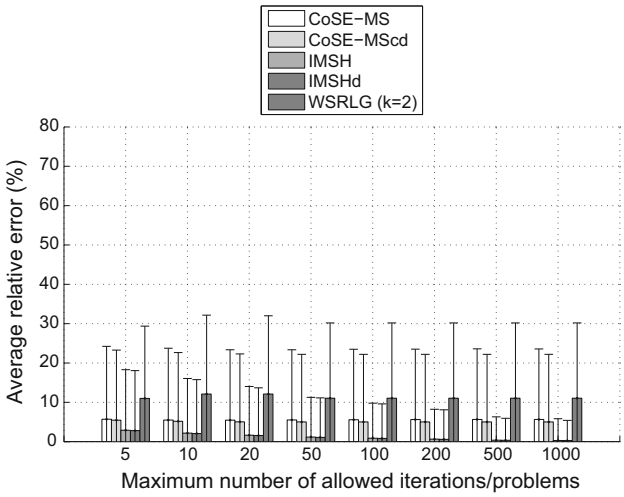

Fig. 3 Relative error (%) of the sub-optimal solutions found for $k = 2$ by CoSE-MS, CoSE-MScd, IMSH, IMSHd, and WSRLG for $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1{,}000$

optimal solution were added and divided by total number of sub-optimal solutions in that network, resulting in the point estimate for a given network of the error of the sub-optimal solutions. The average error in Figs. 3, 6 and 9 is the average of the ten values corresponding to the ten SRLG distributions.

The sub-optimal solutions of IMSH and IMSHd present the smaller average relative error, always below 5 %, and below 1.2 % for $i_{\max} \geq 50$, as can be seen in Fig. 3. The average relative error of CoSE-MScd is slightly smaller than the corresponding average of CoSE-MS, and about 5 % for $i_{\max} \geq 50$. Although the width of the confidence intervals of the relative error is quite wide for all the heuristics, leading to significant interval overlapping, WSRLG is the algorithm with worst average relative error of the solutions.

In Fig. 1 $(k = 2)$ the number of solutions found for $i_{\max} = 50$ is over 99 % for IMSH, IMSHd, COSE-MS and CoSE-MScd. In Fig. 4 $(k = 3)$ the number of solutions found is smaller than the number in Fig. 1, as would be expected. The number of solutions found by kIMSHd and kCoSE-MScd, when a set of $k = 3$ node and SRLG-disjoint solutions are sought, is nevertheless quite high and significantly higher than for WSRLG, especially for $i_{\max} \geq 50$. The heuristics kIMSHd and kCoSE-MScd found 96 and 91 % of the existing solutions, respectively, when $i_{\max} = 50$, while WSRLG did not reach 80 % of the existing solutions. In the case of kIMSHd the number of solutions found keeps increasing with $i_{\max}$ until 99 % for $i_{\max} = 1{,}000$. However the number of optimal solutions found is in average between 55 and 60 % both for kIMSHd and kCoSE-MScd, and their performance does not seem to improve significantly for $i_{\max} \geq 100$, as can be seen in Fig. 5. In fact, for $i_{\max} = 50$, kIMSHd and kCoSE-MScd found in average 58.5 and 57 % of the existing optimal solutions, respectively. In that figure kIMSHd presents a slightly higher average value of optimal solutions than kCoSE-MSCd, but their confidence intervals partially overlap.

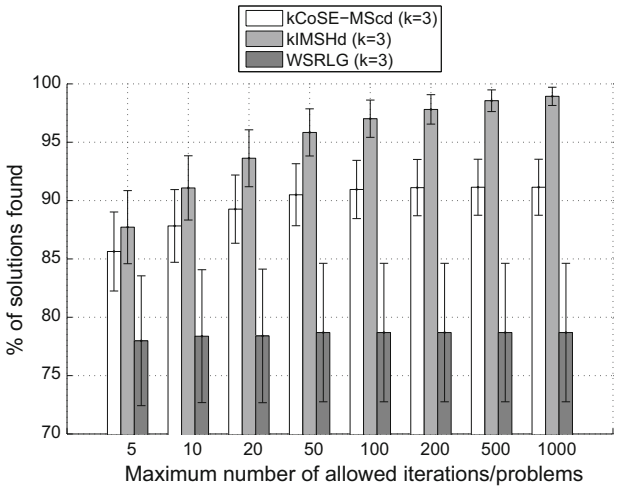

Fig. 4 Average number (%) of solutions found by kCoSE-MScd, kIMSHd, and WSRLG when $k = 3$ for $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1{,}000$

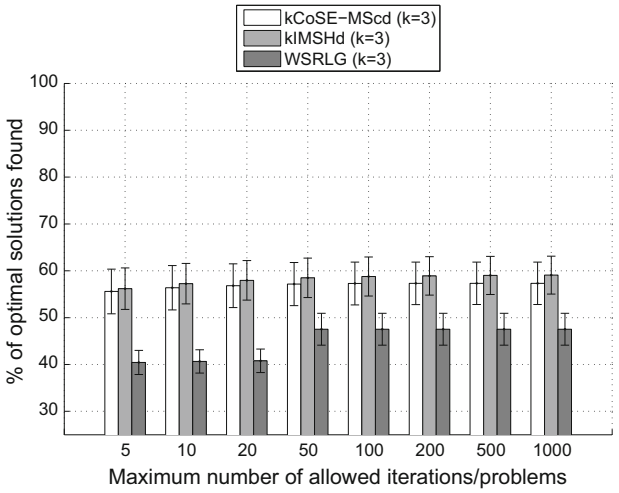

Fig. 5 Average number (%) of optimal solutions found by kCoSE-MScd, kIMSHd, and WSRLG when $k = 3$ for $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1{,}000$

The quality of the sub-optimal solutions for $k = 3$ is shown in Fig. 6. It can be observed, that the average error of kCoSE-MScd and of kIMSHd is very similar, and in the intervals 7.0–7.4 % and 7.2–7.9 %, respectively, while the average error of the sub-optimal solutions of WSRLG is close to 14 %.

In Figs. 7 and 8 the number of solutions found and the number of optimal solutions when the set size is $k = 4$ are presented. It can be seen that IMSHd still manages to find, in average, solutions for over 96 % of the node pairs, but now it requires 200 iterations instead of 50 as in Fig. 4, when $k = 3$; kCoSE-MScd finds over 80 % of the solutions for $i_{\max} \geq 10$, and about 83 % for $i_{\max} \geq 100$. The

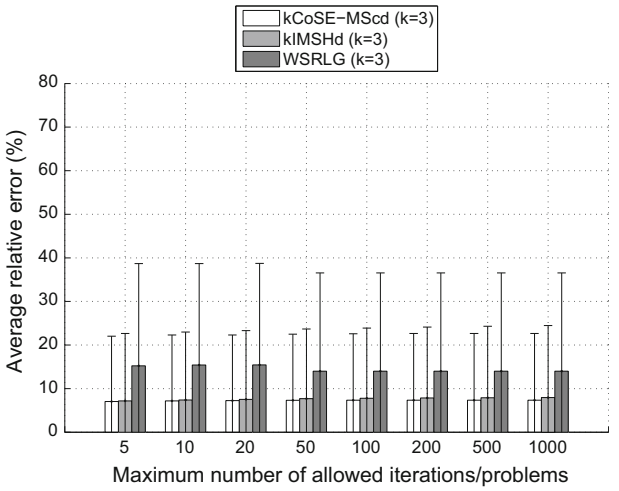

Fig. 6 Relative error (%) of the sub-optimal solutions found by kCoSE-MScd, kIMSHd, and WSRLG when $k = 3$ for $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1{,}000$

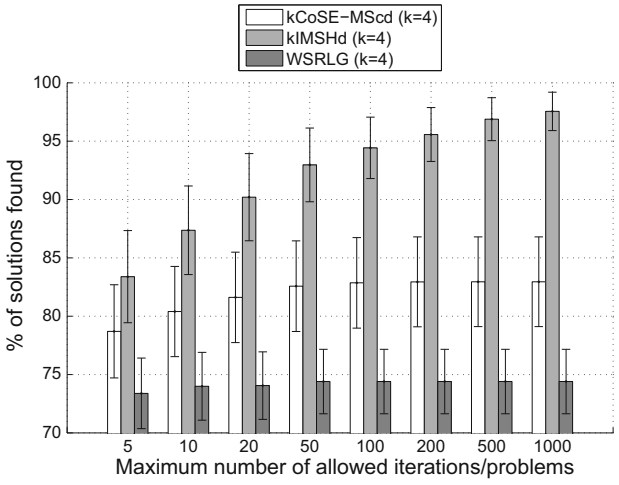

Fig. 7 Average number (%) of solutions found by kCoSE-MScd, kIMSHd, and WSRLG when $k = 4$ for $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1{,}000$

percentage of the existing optimal solutions found by the heuristics has also decreased, but kIMSH still manages to find 50 % of those solution, for $i_{\max} = 50$. kCoSE-MScd finds 48 % of the optimal solutions for $i_{\max} = 50$. WSRLG only finds <75 % of the existing solutions and about 36 % of the existing optimal solutions (see Figs. 7, 8), and is clearly the heuristic with worst performance, concerning these metrics.

The average relative error of the sub-optimal solutions in Fig. 9 is now 12.7–13.2 % for kCoSE-MScd, 13.4–15.0 % for kIMSHd and almost 40 % when

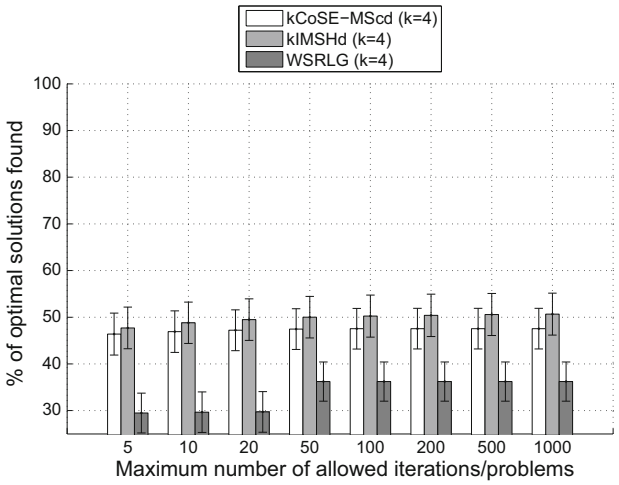

Fig. 8 Average number (%) of optimal solutions found by kCoSE-MScd, kIMSHd, and WSRLG when $k = 4$ for $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1{,}000$

$i_{\max} = 5, 10, 20$ and around 36 % when $i_{\max} \geq 50$ for WSRLG. The width of the confidence intervals is also larger for WSRLG.

Considering the fixed number of tested networks, increasing $i_{\max}$ would not narrow the confidence intervals for the relative error, as can be seen in Figs. 3, 6 and 9, except for IMSH (or IMSHd) for $k = 2$ (where the number of sub-optimal solutions significantly decreases with $i_{\max}$)—and these are the heuristics with smaller relative error. For $k = 3, 4$ the relative error of some of the solutions will remain large, regardless of increasing $i_{\max}$, as the average number of sub-optimal solutions increases slightly or remains fairly unchanged for $i_{\max} \geq 50$.

So, although the number of optimal solutions found by kIMSHd and kCoSE-MScd (for $k = 3, 4$) is not as high as for IMSHd and CoSE-MScd (for $k = 2$), the total number of solutions found is still very high and the average relative error of the sub-optimal solutions is acceptable (7–15 %). Namely, both kIMSHd and kCoSE-MScd perform significantly better than WSRLG, regarding the total number of solutions and their accuracy (number of optimal solutions and average relative error of the sub-optimal solutions).

6.3 CPU Time

Firstly the relative performance of the heuristics regarding the CPU time in the PCE using a shared library will be presented and discussed.

In Table 1 (and the following tables) the values in the line with $i_{\max} = 50$ are emphasized because CoSE-MScd performance does not seem to improve significantly for $i_{\max} > 50$ and it also corresponds to the maximum number of iterations of WSRLG.

From Table 1 it can be seen that IMSHd uses slightly more time than IMSH. That can be considered the cost for IMSHd tending to obtain in average more solutions

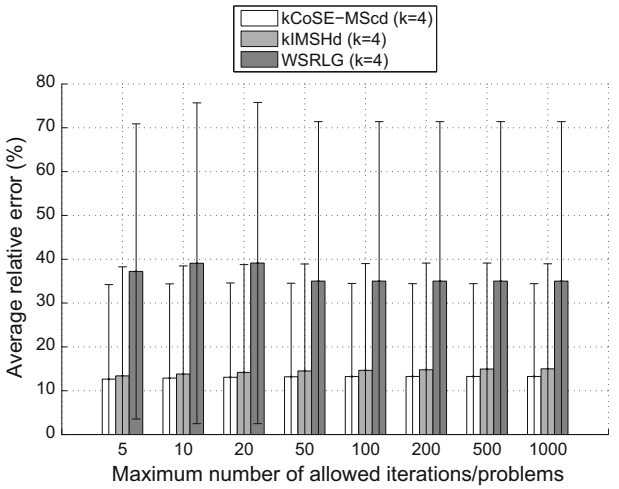

Fig. 9 Relative error (%) of the sub-optimal solutions found by kCoSE-MScd, kIMSHd, and WSRLG when $k = 4$ for $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1{,}000$

and more optimal solutions than IMSH. In the case of CoSE-MScd, the slight increase in CPU time due to the use of MSHd instead of MSH (observed in IMSHd versus IMSH) is largely compensated by the smaller number of auxiliary problems generated, resulting from the new calculation of the set of Conflicting SRLG, as explained in Sect. 5.2 (and illustrated in line 31 of auxiliary algorithm AllPairs). In fact from Table 1 the average number of auxiliary problems solved by CoSE-MScd is <100, because the CPU time is stable for $i_{\max} \geq 100$, in contrast with CoSE-MS, where the CPU time grows with $i_{\max}$.

WSRLG uses less CPU time then IMSHd, except for $i_{\max} = 20, 50$, and for $i_{\max} = 50$ WSRLG requires over 25 % more CPU time than IMSHd. CoSE-MScd is the heuristic with better performance regarding CPU time. Requiring <25 ms per node pair for $i_{\max} \geq 50$) in the used PCE, CoSE-MScd is adequate for calculating node and SRLG-disjoint path pairs in the control plane of a GMPLS network.

Table 1 Average CPU (seconds) time for $k = 2$ in a path computation element (330 MHz, 128 MB), for a total of 1,000 node pairs randomly chosen in each network

$i_{\max}$	IMSH	CoSE-MS	IMSHd	CoSE-MScd	WSRLG
5	75.5 ± 2.1	17.3 ± 1.6	77.6 ± 1.5	12.4 ± 0.9	31.2 ± 0.4
10	89.8 ± 2.3	27.2 ± 2.8	93.5 ± 1.7	16.3 ± 1.3	60.0 ± 0.8
20	117.4 ± 2.7	45.5 ± 5.3	124.7 ± 2.2	21.2 ± 2.1	118.3 ± 1.7
50	**195.9 ± 4.4**	**95.1 ± 11.7**	**214.6 ± 3.9**	**23.9 ± 2.9**	**293.7 ± 5.2**
100	319.0 ± 8.2	169.3 ± 20.9	356.7 ± 8.6	24.1 ± 3.2	293.7 ± 5.2
200	551.2 ± 17.8	302.0 ± 36.2	624.2 ± 19.8	24.1 ± 3.1	293.7 ± 5.2
500	1,207.9 ± 50.8	644.7 ± 78.9	1,385.9 ± 57.1	24.1 ± 3.1	293.7 ± 5.2
1,000	2,238.8 ± 103.5	1,166.8 ± 136.4	2,590.4 ± 120.9	24.2 ± 3.1	293.7 ± 5.2

Table 2 Average CPU time (seconds) for $k = 3$ in a path computation element (330 MHz, 128 MB), for a total of 1,000 node pairs randomly chosen in each network

i_{max}	kIMSHd	kCoSE-MScd	WSRLG
5	114.3 ± 3.7	56.0 ± 0.7	33.6 ± 0.3
10	167.9 ± 3.1	104.8 ± 1.2	64.6 ± 0.5
20	255.6 ± 3.7	193.3 ± 2.2	126.9 ± 1.1
50	**528.9 ± 7.2**	**411.0 ± 5.8**	**313.7 ± 2.8**
100	997.3 ± 13.1	682.3 ± 16.0	313.6 ± 2.8
200	1,957.8 ± 28.0	1,006.5 ± 37.7	313.6 ± 2.8
500	4,926.4 ± 64.0	1,314.1 ± 76.3	313.6 ± 2.8
1,000	10,015.1 ± 138.9	1,384.3 ± 84.3	313.7 ± 2.8

Table 3 Average CPU time (seconds) for $k = 4$ in a path computation element (330 MHz, 128 MB), for a total of 1,000 node pairs randomly chosen in each network

i_{max}	kIMSHd	kCoSE-MScd	WSRLG
5	138.1 ± 3.9	76.4 ± 0.8	34.2 ± 0.3
10	207.5 ± 2.4	145.2 ± 1.5	65.7 ± 0.6
20	337.5 ± 4.2	269.9 ± 2.7	128.9 ± 1.3
50	**736.5 ± 10.5**	**577.5 ± 7.6**	**318.6 ± 3.2**
100	1,416.3 ± 20.4	960.8 ± 21.6	318.5 ± 3.2
200	2,802.8 ± 42.5	1,416.2 ± 50.5	318.5 ± 3.2
500	7,088.4 ± 112.0	1,841.6 ± 103.0	318.5 ± 3.2
1,000	14,271.9 ± 263.9	1,935.3 ± 112.5	318.5 ± 3.2

From Tables 2 and 3 it can be seen that WSRLG uses less CPU time than kIMSHd or kCoSE-MScd, and does not seem to use much more CPU time when k (the size of the set) goes from 3 to 4. Although WSRLG used less CPU than kIMSHd or kCoSE-MScd, as shown in Sect. 6.2, it obtains significantly fewer solutions and significantly fewer optimal solutions than kIMSHd or kCoSE-MScd, and the sub-optimal solutions of WSRLG also present the average largest relative error.

The CPU in Tables 2 and 3 are not adequate for the control plane (for $i_{max} \geq 50$), unless newer technology PCE with higher capabilities can be used. However the CPU time is adequate for answering the request for a protected end-to-end path (considering SRLG) in the management plane. In this context kIMSHd should be the preferred heuristic, because for $i_{max} = 50$ the number of solutions is 96 and 93 % for $k = 3$ and $k = 4$, respectively, and the relative error of the sub-optimal solutions is not too high.

In Figs. 10, 11 and 12 the lines present CPU times corresponding to in Tables 1, 2 and 3, respectively. The y axis values indicate the estimated number of pairs (in %) for which an optimal solution was obtained for 1,000 random node pairs. Notice the logarithmic scale in the x axis with the total CPU time in the PCE for the considered 1,000 random node pairs. The first point in each curve corresponds to the CPU time and number of pair with optimal solutions after the first five iterations

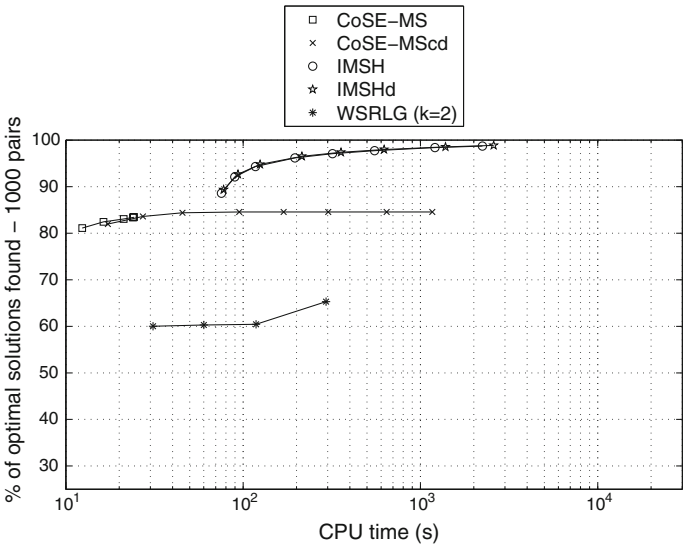

Fig. 10 Considering 1,000 random pairs, estimated average number (%) of pairs with optimal solutions found for $k = 2$ by CoSE-MS, CoSE-MScd, IMSH, IMSHd, and WSRLG and corresponding total CPU time in the PCE for $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1{,}000$

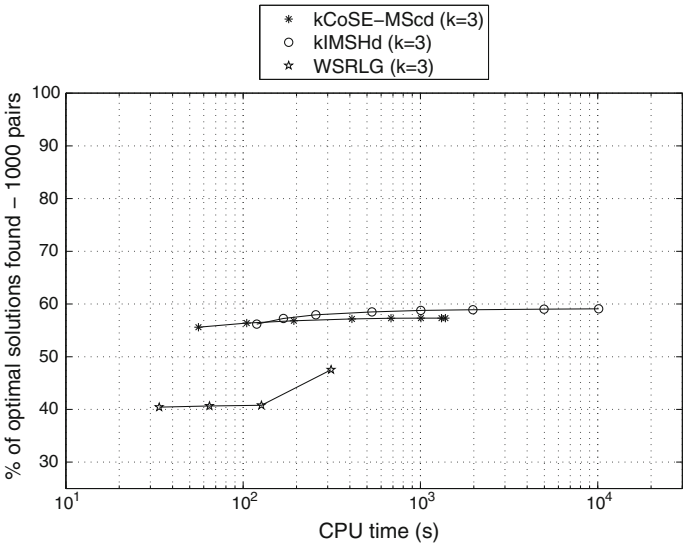

Fig. 11 Considering 1,000 random pairs, estimated average number (%) of pairs with optimal solutions found by kCoSE-MScd, kIMSHd, and WSRLG when $k = 3$ and corresponding total CPU time in the PCE for $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1{,}000$

(and similarly for the following points corresponding to the next considered values for $i_{\max}$). It can be seen that WSRLG has apparently less points than the other heuristics, but this explained by the fact that after 50 iterations the CPU time remains unchanged. Similar effect can be observed for CoSE-MScd and for $k = 2$ in

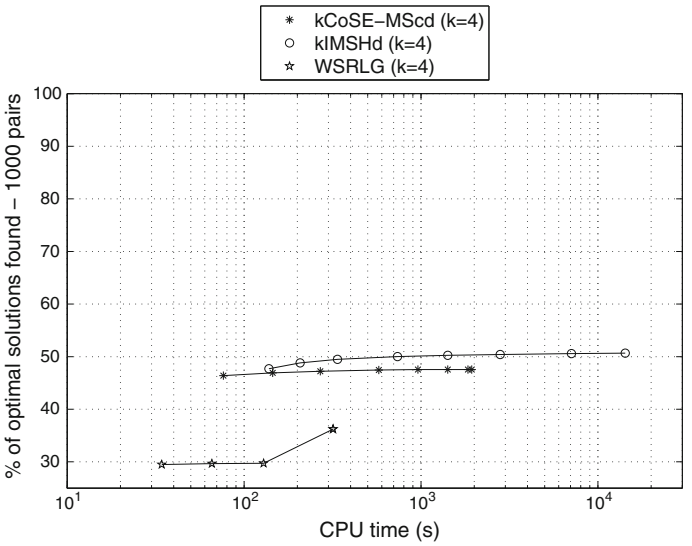

Fig. 12 Considering 1,000 random pairs, estimated average number (%) of pairs with optimal solutions found by kCoSE-MScd, kIMSHd, and WSRLG when $k = 4$ and corresponding total CPU time in the PCE for $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1{,}000$

Table 4 Average CPU time (seconds) for $k = 2$ in a Desktop (i7 CPU 870 @2.93 GHz, 3.6 GB), for a total of 5,000 node pairs randomly chosen in each network and $i_{\max} = 50$

k	IMSHd	CoSE-MScd	WSRLG	CPLEX
2	29.8 ± 0.5	18.3 ± 1.9	51.4 ± 0.4	1,713.3 ± 108.2

Table 5 Average CPU time (seconds) for $k = 3, 4$ in a Desktop (i7 CPU 870 @2.93 GHz, 3.6 GB), for a total of 5,000 node pairs randomly chosen in each network, and $i_{\max} = 50$

k	kIMSHd	kCoSE-MScd	WSRLG	CPLEX
3	82.9 ± 0.7	70.7 ± 1.3	55.3 ± 0.5	1,705.9 ± 77.1
4	113.9 ± 1.0	96.0 ± 1.4	56.3 ± 0.5	2,122.7 ± 183.8

Fig. 10. It can be seen that for the same CPU time IMSH/IMSHd and kISMH perform better than any other heuristic, particularly for CPU times above 110 s.

As seen in Table 4, on the Desktop the CPU times of both IMSHd and CoSE-MScd, considering $i_{\max} = 50$, are significantly smaller then the ones required by CPLEX. Of course this comes at the cost of finding only about 96 % of the optimal solutions.

For $k = 3, 4$, and considering $i_{max} = 50$, the CPU times of the heuristics, in Table 5, are still significant smaller then the CPU required by CPLEX.

The CPU time required by the CPLEX solver is >1 s per node pair, in the Desktop used, for $k = 2, 3, 4$. However note that in a PCE the CPLEX solver is not an option, and hence the practical interest of the developed heuristics.

7 Conclusion

The concept of SRLG allows an upper layer to establish a protected connection, selecting an AP and a BP which should be SRLG-disjoint. Routing protocols in GMPLS, using distributed SRLG information, can calculate paths avoiding certain SRLGs. For single SRLG failure end-to-end SRLG-disjoint paths can be calculated, but for ensuring against multiple SRLG failures a set of end-to-end SRLG-disjoint paths should be used. Two heuristics, the Conflicting SRLG-Exclusion Min Sum (CoSE-MS) and the Iterative Modified Suurballes's Heuristic (IMSH), for calculating SRLG-disjoint path pairs, which use the Modified Suurballes's Heuristic (MSH), were reviewed and new versions (CoSE-MScd and IMSHd) were proposed which may improve the number of optimal solutions found. In the case of IMSHd this is achieved at the cost of a slight increase in CPU cost; in the case of CoSE-MScd the modification in the calculation of the conflicting SRLG set resulted in less problems to solve and in a significant decrease in CPU time, which makes it adequate for use in the control plane of a GMPLS network.

A generalization of MSH for obtaining a set of k node and SRLG-disjoint paths, given a set of $k-1$ seed paths, which we designate as kMSH was introduced. The heuristics kCoSE-MScd and kIMSHd were then proposed for calculating a set of node and k SRLG-disjoint paths, seeking to minimize its total cost. To the best of our knowledge these heuristics are a first proposal for seeking a set of $k\,(k > 2)$ node and SRLG-disjoint paths of minimal additive cost; the two heuristics have a similar structure, but the first uses CoSE-MScd to collect a seed set of node and SRLG-disjoint path pairs and the second uses IMSHd for that same purpose.

The performance of the proposed heuristics was evaluated using a real network, where SRLGs were randomly defined. The number of solutions found, the percentage of optimal solutions and the relative error of the sub-optimal solutions were presented and discussed. The quantity and quality of the solutions obtained using kIMSHd and kCoSE-MScd is significantly better than the ones obtained by WSRLG, although the later uses less CPU time.

For $k = 2$ CoSE-MScd is a good compromise solution for use in the control plane of a GMPLS network. But if a PCE with higher performance becomes available, IMSHd (with $i_{max} = 50$) could be a more accurate alternative.

For $k = 3, 4$, considering the number of allowed iterations equal to 50, and given the percentage of node pairs for which was possible to obtain a solution and the relative error of the sub-optimal solutions, the IMSHd is an effective practical resolution procedure which provides a good compromise between CPU time and

the solution quality, for calculating an optimal/sub-optimal set of node and SRLG-disjoint paths in the context of a request in the management plane using a PCE.

Acknowledgments The author acknowledges financial support through project QREN 23301 PANORAMA II, co-financed by European Union's FEDER through "Programa Operacional Factores de Competitividade" (POFC) of QREN (FCOMP-01-0202-FEDER-023301) by the Portuguese Foundation for Science and Technology under project grant PEst-OE/EEI/UI308/2014 and by Project PERGS of Portugal Telecom Inovação. We thank the anonymous referees for their comments and suggestions, which contributed to improve the paper.

Appendix 1: Example Illustrating MSH

In the network in Fig. 13 there are three SRLGs, marked as ellipses: $g_1 = \{(1,3),(3,1),(1,4),(4,1)\}$, $g_2 = \{(3,7),(7,3),(4,7),(7,4)\}$ and $g_3 = \{(5,8),(8,5),(6,8)\}$. The label of each arc corresponds to its cost.

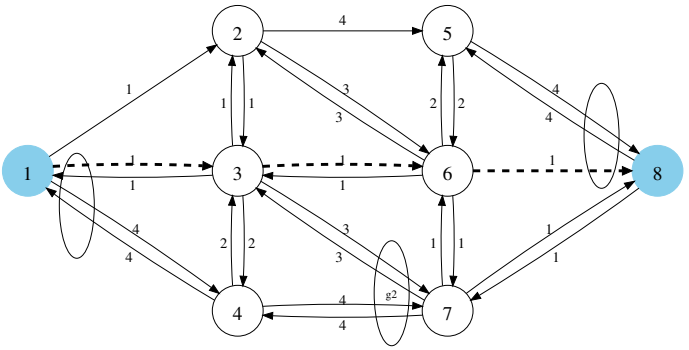

Fig. 13 Directed network G, where $s = 1$ and $t = 8$. The seed path is $p_1 = \langle 1,3,6,8 \rangle$, and three different SRLG are marked

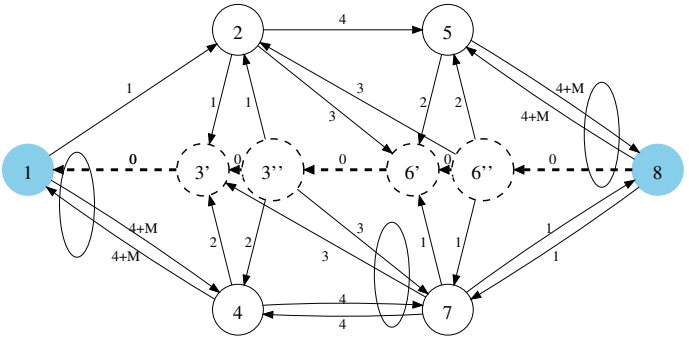

Fig. 14 Directed network G', after dividing the network using the MSH transformation

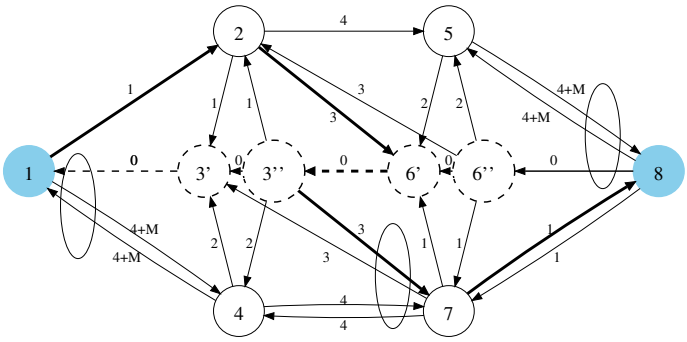

Fig. 15 Shortest path $q'_1 = \langle 1, 2, 6', 3'', 7, 8 \rangle$ in G'

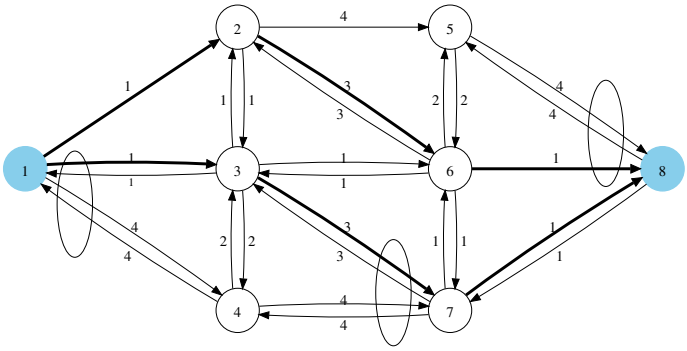

Fig. 16 Solution $(p, q) = (\langle 1, 2, 6, 8 \rangle, \langle 1, 3, 7, 8 \rangle)$

The original network graph G', and the shortest path from node 1 to node 8 is shown in Fig. 13. The directed network G', after the network transformation of the MSH (as described in Sect. 4.1) is shown in Fig. 14. In Fig. 15 the interlacing arc (3, 6) is removed and the solution is shown in Fig. 16.

Appendix 2: Auxiliary Heuristic AllPairs

The heuristic kCoSE-MScd requires a version of CoSE-MS that stores all node and SRLG-disjoint pairs discovered, during the $i_{\max}$ iterations or until the stack of problems is empty. This task is performed by the heuristic AllPairs.

Data: Digraph $G = (V, A)$; source node s; target node t; arc cost $l(v_i, v_j)$; $R(v_i, v_j)$, the SRLG associated with each arc $(v_i, v_j) \in A$; maximal number of iterations $i_{\max}$.
Result: Stack of node and SRLG-disjoint paths P

```
1  i ← 0
2  Δ = (min_{(i,j)∈A} l(i,j))/(2|V|)                    // Assumes min_{(i,j)∈A} l(i,j) > 0
3  P ← ∅                  // Empty stack of node and SRLG-disjoint path pairs P
4  S ← ∅                                                 // Empty stack of problems
   // (p,q) is presently the best solution found of cost c_(p,q)
5  (p,q) ← (∅,∅)  // (p,q) is presently the best solution found of cost ∞
6  P_0 ← (∅,∅,∅), push(S, P_0)                          // First problem in stack S
7  i ← 0                                          // Counting the problems solved
8  while ¬empty(S) ∧ i < i_max do
9      P_c(I_c, E_c, H_c) ← top(S)                        // P_c is the present problem
10     pop(S)                              // Removes P_c from the top of the stack
11     i ← i + 1                          // Updates the counter of problems solved
12     p_c ← shortest path from s to t in the present problem
13     if c_{p_c} ≠ ∞       // If a shortest path exists in the present problem
14     then
15         if P_c = P_0 then
16             (p'_c, p''_c) ←MBH(G, s, t, l, R, p_c)
17         end
18         else
19             (p'_c, p''_c) ←MSHd(G, s, t, l, R, p_c, Δ)
20         end
21         X ← R_{p'_c} ∩ R_{p''_c}
22         if X = ∅ ∧ (p'_c, p''_c) ≠ (∅,∅)              // p'_c and p''_c are SRLG-disjoint
23         then
24             push(S_p, (p'_c, p''_c))  // Stores node and SRLG-disjoint path pair
25         end
26         else
27             if P_c = P_0 ∨ (X = ∅ ∧ (p'_c, p''_c) = (∅,∅)) then
                   // If P = P_0 or there is no node-disjoint path pair
28                 T_c ←SRLG_Exclusion(I_c, p_c)
29             end
30             else
                   // There is a node-disjoint but not SRLG-disjoint path
                      pair
31                 T_c ← (X ∩ R_{p_c}) \ I_c
32             end
               // Let T_c be the set {g_1, g_2, ..., g_{|T_c|}}
33             H ← E_c ∪ H_c
34             P_1(I_1, E_1, H_1) ← P(∅, {g_1}, H)
35             push(S, P_1)
36             j ← 2
37             while j ≤ |T_c| do
38                 P_j(I_j, E_j, H_j) ← P(I_{j−1} ∪ E_{j−1}, {g_j}, H)
39                 push(S, P_j)
40                 j ← j + 1
41             end
42         end
43     end
44 end
```

Heuristic AllPairs: Auxiliary of kCoSE-MScd, returns a stack of node and SRLG-disjoin path pairs.

Given a seed path of problem P_c, calculated in the network where the arcs affected by the SRLGs $E_c \cap H_c$, have been removed, the MBH or MSH seek to

obtain an SRLG path pair of min-sum cost. If no such pair is found, the conflicting SRLG set must be found. The set T_c is calculated as described in Sect. 4.1. The function SRLG_EXCLUSION (I_c, p_c) in line 28 of AllPairs, corresponds to algorithm "Algorithm. Finding a conflicting SRLG set for a given AP p from node s to node t" in [26] and is now used only when no node-disjoint path pair can be found.

In line 31 of AllPairs is the new procedure for obtaining the conflicting SRLG set T_c, used when a node disjoint path pair, which is not SRLG-disjoint, exists (as described in Sect. 4.1).

Appendix 3: Illustrating the Failure of the IMSH Optimal Condition

In [29] the following proof is presented of the optimality of the best current path pair (p, q), which we reproduce here using our notation.

> Let (p, q) be the current optimal SRLG diverse path pair found and its cost be $c_{(p,q)}$. In the i-th iteration, let c_{p_i} be the cost of the shortest path computed using Yen's algorithm [38]. Let (p_i', p_i'') be the SRLG diverse path pair computed using modified Suurballe's heuristic, if such a path pair exists. Let (p_i', p_i'') be more optimal than the current optimal (p, q), i.e,
>
> $$c_{p_i'} + c_{p_i''} < c_{(p,q)} \tag{14}$$
>
> Now $c_{p_i'}, c_{p_i''} \geq c_{p_i}$. Since, without loss of generality, if $c_{p_i'} < c_{p_i}$ the optimal SRLG path pair must have already been computed using p_i' as the seed path. Therefore,
>
> $$2c_{p_i} \leq c_{p_i'} + c_{p_i''} \tag{15}$$
>
> From Eqs. 14 and 15 we get,
>
> $$\begin{aligned} 2c_{p_i} &< c_{(p,q)} \\ c_{p_i} &< c_{(p,q)}/2 \end{aligned}$$
>
> Therefore if the cost of the current seed path in the i-th iteration is greater than or equal to $c_{(p,q)}/2$ then the optimal SRLG diverse path pair is (p, q).

The problem with this proof, is in the statement "*Since, without loss of generality, if* $c_{p_i'} < c_{p_i}$ *the optimal SRLG path pair must have already been computed*

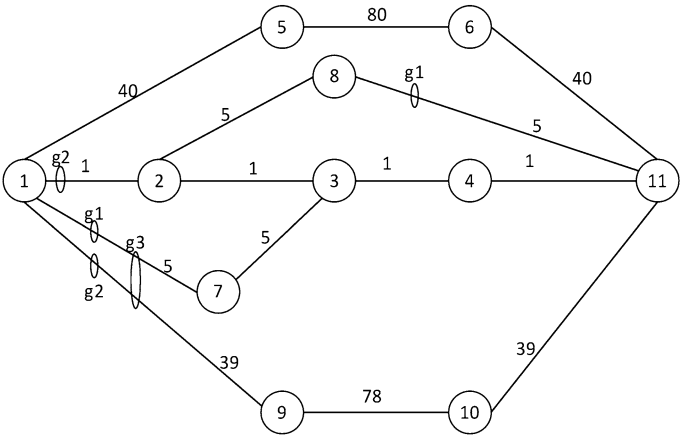

Fig. 17 Network for illustrating the failure of the optimal condition in [29]

using p'_i as the seed path." which does not hold for generic randomly generated SRLG.

Let $p'_i = p_j, j < i$, be the shortest of the current pair obtained in the i-th iteration ($c_{p'_i} \leq c_{p''_i}$). When p_j was used as seed path it may have resulted in a path pair which is not SRLG-disjoint, due to the interlacing removal. Hence $p'_i = p_j, j < i$ may appear later, in an SRLG-disjoint path pair resulting from using a seed path p_i, and this contradicts the previous statement. We next will illustrate, using an example that, in networks with randomly generated SRLGs, that this is the reason why the proof fails.

In the Fig. 17 an undirected network is represented. The SRLGs are: $g_1 = \{(1,7),(8,11)\}$, $g_2 = \{(1,2),(1,9)\}$ and $g_3 = \{(1,7),(1,9)\}$. Initially the best solution is $(p,q) = (\emptyset,\emptyset)$ of cost ∞. Algorithm IMSH would have the following iterations:

Iteration 1: $p_1 = \langle 1,2,3,4,11\rangle$ of cost 4, $R_{p_1} = \{g2\}$. The shortest path in the modified network is $q'_1 = \langle 1,7,3,2,8,11\rangle$ of cost 20 (in G'). These paths are SRLG-disjoint, but an interlacing exists, and after removing that interlacing the resulting path pair is:

- $p'_1 = \langle 1,2,8,11\rangle$ of cost 11, $R_{p'_1} = \{g_1, g_2\}$;
- $p''_1 = \langle 1,7,3,4,11\rangle$ of cost 12, $R_{p''_1} = \{g_1, g_3\}$.

Because $R_{p'_1} \cap R_{p''_1} = \{g_1\}$ the path pair is not SRLG-disjoint and hence is not admissible.

Iteration 2: $p_2 = \langle 1,2,8,11\rangle$ of cost 11, $R_{p_2} = \{g_1, g2\}$. The shortest path in the modified network is $q'_2 = \langle 1,5,6,11\rangle$ of cost 160 (in G'), $R_{q'_2} = \emptyset$. There is no interlacing, and the resulting path pair is:

- $p'_2 = \langle 1,2,8,11\rangle$ of cost 11, $R_{p'_2} = \{g_1, g2\}$;

- $p_2'' = \langle 1, 5, 6, 11\rangle$ of cost 160, $R_{p_2''} = \emptyset$;

which is SRLG-disjoint. The best solution is updated: $(p, q) = (\langle 1, 2, 8, 11\rangle, \langle 1, 5, 6, 11\rangle)$, and $c_{(p,q)} = 171$.

Iteration 3: $p_3 = \langle 1, 7, 3, 4, 11\rangle$ of cost 12, $R_{p_3} = \{g_1, g3, \}$. The shortest path in the modified network is $q_3' = \langle 1, 5, 6, 11\rangle$ of cost 160, $R_{q_3'} = \emptyset$. There is no interlacing, and the resulting path pair is:

- $p_3' = \langle 1, 7, 3, 4, 11\rangle$ of cost 12, $R_{p_2} = \{g_1, g2\}$;
- $p_3'' = \langle 1, 5, 6, 11\rangle$ of cost 160, $R_{p_2} = \emptyset$;

which is SRLG-disjoint. The best solution is not updated because $c_{(p_3', p_3'')} = 172$ is greater than $c_{(p,q)} = 171$.

Iteration 4: $p_4 = \langle 1, 7, 3, 2, 8, 11\rangle$ of cost 21, $R_{p_4} = \{g_1, g3\}$. The shortest path in the modified network is $q_4' = \langle 1, 2, 3, 4, 11\rangle$ of cost 3 (in G'), $R_{q_4'} = \{g_2\}$. These paths are SRLG-disjoint, but an interlacing exists, and after removing that interlacing the resulting path pair is:

- $p_4' = \langle 1, 7, 3, 4, 11\rangle$ of cost 12, $R_{p_4'} = \{g_1, g_3\}$;
- $p_4'' = \langle 1, 2, 8, 11\rangle$ of cost 160, $R_{p_4''} = \{g_1, g_2\}$;

Because $R_{p_4'} \cap R_{p_4''} = \{g_1\}$ the path pair is not SRLG-disjoint and hence is not admissible.

Iteration 5 $p_5 = \langle 1, 9, 10, 11\rangle$ of cost 156. Because $c_{(p,q)} \leq 2c_{p_5}$ $(171 \leq 2 \times 156)$ the algorithm would end considering that the best solution found so far, of cost 171, is the optimal solution.

It can be easily seen that at the 6-th iteration, with $p_6 = \langle 1, 5, 6, 11\rangle$ of cost 160, $R_{p_6} = \emptyset$, in the modified graph $q_6' = \langle 1, 2, 3, 4, 11\rangle$ of cost 4 (which coincides with p_1), $R_{p_1} = R_{q_6'} = \{g2\}$, would result in the path pair:

- $p_6' = \langle 1, 2, 3, 4, 11\rangle$ of cost 4, $R_{p_6'} = \{g2\}$
- $p_6'' = \langle 1, 5, 6, 11\rangle$ of cost 160, $R_{p_6''} = \emptyset$;

which is SRLG-disjoint an has cost 164. In this case $c_{p_6'} < c_{p_6}$ and the optimal solution was not found when the seed path was $p_1 = p_6'$.

However, in a network where the SRLGs are strictly local (that is all the edges in each SRLG have the same node in common), if p_i and q_i' are SRLG-disjoint, the interlacing removal will never result in a non SRLG-disjoint solution (as in iteration 1 of this example).

References

1. Mouftah, H.T., Ho, P.-H.: Optical networks: arquitecture and survivability. Kluwer Academic Publishers, Dordrecht (2003)
2. Tapolcai, J., Pin-Han Ho, D., Verchere, T.Cinkler: A new shared segment protection method for survivable networks with guaranteed recovery time. IEEE Trans. Reliab. **57**(2), 272–282 (2008)

3. Jaumard, B., Nahar Bhuiyan, N., Sebbah, S., Huc, F., Coudert , D.: A new framework for efficient shared segment protection scheme for WDM networks. In: International Conference on High Performance Switching and Routing (HPSR), 2010, pp. 189–196. (2010)
4. Medhi, D., Sankarappan, S.: Impact of a transmission facility link failure on dynamic call routing cicuit-switched networks under various circuit layout policies. J. Netw. Syst. Manag. **1**(2), 143–169 (1993)
5. Rajagopalan, B., Pendarakis, D., Saha, D., Ramamoorthy, R.S.: IP over optical networks: architectural aspects. IEEE Commun. Mag. **38**(9), 94–102 (2000)
6. Vasseur, J.-P., Pickavet, M., Demeester, P.: Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS. Elsevier, Amsterdam (2004)
7. Braden, R., Zhang, L., Berson, S., Herzog, S., Jamin, S.: Resource reservation protocol (RSVP—version 1 functional specification. IETF RFC 2205). (1997)
8. Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., Swallow, G.: RSVP-TE: extensions to RSVP for LSP tunnels. IETF RFC 3209. (2001)
9. Zhang, F., Li, D., Gonzalez de Dios, O., Margaria, C.: RSVP-TE extensions for collecting SRLG information. IETF Draft. (2014)
10. Kompella, K., Rekhter, Y.: Routing extensions in support of generalized multi-protocol label switching (GMPLS). IETF RFC 4202. (2005)
11. Farrel, A., Vasseur, J.-P., Ash, J.: A path computation element (PCE)-based architecture. ITEF RFC 4655. (2006)
12. Saturio, M. C., López, V., Dios, Ó. G., del Nuevo, F. M., Fernández-Palacios, J. P.: Implementation and assessment of pre-reservation mechanism for PCE environments. J. Netw. Syst. Manag. **22**(3), 488–508 (2014)
13. Suurballe, J.W., Tarjan, R.E.: A quick method for finding shortest pairs of disjoint paths. Networks **14**(2), 325–336 (1984)
14. Bhandari, R.: Survivable Networks, Algorithms for Diverse Routing. Kluwer Academic Publishers, Norwell, MA (1999)
15. Xu, D., Chen, Y., Xiong, Y., Qiao, C., He, X.: On finding disjoint paths in single and dual link cost networks. In: IEEE INFOCOM 2004. Hong Kong (2004)
16. Kodialam, M., Lakshman, T.V.: Dynamic routing of bandwidth guaranteed tunnels with restoration. IEEE INFOCOM **2000**, 902–911 (2000)
17. Laborczi, P., Tapolcai, J., Ho, P.-H., Cinkler, T., Recski, A., Mouftah, H.T.: Algorithms for asymmetrically weighted pair of disjoint paths in survivable networks. In: Cinkler, T. (ed.) Proceedings of Design of Reliable Communication Networks (DRCN), pp. 220–227. Budapest, Hungary (2001)
18. Gan, Ming-Lee, Liew, Soung-Yue: Effective algorithms for finding optimum pairs of link-disjoint paths in $\alpha + 1$ path protection. Telecommun. Syst. **52**(2), 783–797 (2013)
19. Li, C.L., McCormick, S.T., Simchi-Levi, D.: The complexity of finding two disjoint paths with min–max objective function. Discret. Appl. Math. **26**(1), 105–115 (1990)
20. Gomes, T., Craveirinha, J., Jorge, L.: An effective algorithm for obtaining the minimal cost pair of disjoint paths with dual arc costs. Comput. Oper. Res. **36**(5), 1670–1682 (2009)
21. Rak, J.: *k*-Penalty: a novel approach to find *k*-disjoint paths with differentiated path costs. Commun. Lett. IEEE **14**(4), 354–356 (2010)
22. Hu, J.Q.: Diverse routing in optical mesh networks. IEEE Trans. Commun. **51**(3), 489–494 (2003)
23. Oki, E., Matsuura, N., Shiomoto, K., Yamanaka, N.: A disjoint path selection scheme with shared risk link groups in GMPLS networks. IEEE Commun. Lett. **6**(9), 406–408 (2002)
24. Xu, D., Xiong, Y., Qiao, C., Li, G.: Trap avoidance and protection schemes in networks with shared risk link groups. J. Lightwave Technol. **21**(11), 2683–2693 (2003)
25. Ho, P.-H., Mouftah, H.T.: Shared protection in mesh WDM networks. IEEE Commun. Mag. **42**(1), 70–76 (2004)
26. Rostami, M.J., Khorsandi, S., Khodaparast, A.A.: CoSE: A SRLG-disjoint routing algorithm. In: Proceedings of the Fourth European Conference on Universal Multiservice Networks (ECUMN'07). Toulouse (2007)
27. Gomes, T., Fernandes, L.: Obtaining a SRLG-disjoint path pair of min-sum cost. In: Rak, J., Tipper, D., Walkowiak, K. (eds.) RNDM 2010–2nd International Workshop on Reliable Networks Design and Modeling, colocated with ICUMT 2010. ISBN: 978-I-4244-7283-3, pp. 116–122. Moscow (2010)
28. Gomes, T., Simões, C., Fernandes, L.: Resilient routing in optical networks using SRLG-disjoint path pairs of min-sum cost. Telecommun. Syst. J. **52**(2), 737–749 (2013)

29. Todimala, A., Ramamurthy, B.: IMSH: An iterative heuristic for SRLG diverse routing in WDM mesh networks. In: Proceedings of the 13th International Conference on Computer Communications and Networks, ICCCN'2004, pp. 199–204. (2004)
30. Ho, P.-H., Tapolcai, J., Mouftah, H.T.: On achieving optimal survivable routing for shared protection in survivable next-generation internet. IEEE Trans. Reliab. **53**(2), 216–225 (2004)
31. Datta, Pallab, Somani, Arun K.: Graph transformation approaches for diverse routing in shared risk resource group (srrg) failures. Comput. Netw. **52**(12), 2381–2394 (2008)
32. Lee, H.-W., Modiano, E., Lee, K.: Diverse routing in networks with probabilistic failures. IEEE/ACM Trans. Netw. **18**(6), 1895–1907 (2010)
33. Diaz, O., Xu, F., Min-Allah, N., Khodeir, M., Peng, M., Khan, S., Ghani, N.: Network survivability for multiple probabilistic failures. Commun. Lett. IEEE **16**(8), 1320–1323 (2012)
34. Xu, D., Li, G., Ramamurthy, B., Chiu, A., Wang, D., Doverspike, R.: SRLG-diverse routing of multiple circuits in a heterogeneous optical transport network. In: Proceedings of the 8th International Workshop on the Design of Reliable Communication Networks (DRCN 2011), pp. 180–187. (2011)
35. Xu, Dahai, Li, Guangzhi, Ramamurthy, Byrav, Chiu, Angela, Wang, Dongmei, Doverspike, Robert: On provisioning diverse circuits in heterogeneous multi-layer optical networks. Comput. Commun. **36**(6), 689–697 (2013). Reliable Network-based Services.
36. Habib, M.F., Tornatore, M., De Leenheer, M., Dikbiyik, F., Mukherjee, B.: Design of disaster-resilient optical datacenter networks. J. Lightwave Technol. **30**(16), 2563–2573 (2012)
37. Develder, C., Buysse, J., De Leenheer, M., Jaumard, B., Dhoedt, B.: Resilient network dimensioning for optical grid/clouds using relocation. In: IEEE International Conference on Communications (ICC), 2012, pp 6262–6267. (2012)
38. Yen, J.Y.: Finding the k shortest loopless paths in a network. Manag. Sci. **17**(11), 712–716 (1971)
39. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**, 269–271 (1959)
40. Kuipers, F.A., Korkmaz, T., Krunz, M., Van Mieghem, P.: A review of constraint-based routing algorithms. Technical report, Delft University of Technology, (2002)
41. Clímaco, J., Craveirinha, J.: On OR-based routing approaches for the Internet. Int. Trans. Oper. Res. **18**(3), 295–305 (2011)
42. Martins, E., Pascoal, M.: A new implementation of Yen's ranking loopless paths algorithm. 4OR Q. J. Belg. Fr. Ital. Oper. Res. Soc. **1**(2), 121–134 (2003)
43. Martins, E., Pascoal, M., Santos, J.: Deviation algorithms for ranking shortest paths. Int. J. Found. Comput. Sci. **10**(3), 247–263 (1999)
44. Martins, E., Pascoal, M., Santos, J.: An algorithm for ranking loopless paths. Technical Report 99/007, CISUC. (1999)
45. Clímaco, J., Craveirinha, J., Pascoal, M.: A bicriterion approach for routing problems in multimedia networks. Networks **41**(4), 206–219 (2003)
46. Liu, Chang, Ruan, Lu: p-Cycle design in survivable WDM networks with shared risk link groups (SRLGs). Photonic Netw. Commun. **11**, 301–311 (2006)
47. Zhang, Qingfu, Sun, Jianyong, Xiao, Gaoxi, Tsang, Edward: Evolutionary algorithms refining a heuristic: a hybrid method for shared-path protections in WDM networks under SRLG constraints. IEEE Trans. Syst. Man Cybern. Part B Cybern. **37**(1), 51–61 (2007)
48. Bermond, J.-C., Coudert, D., D'Angelo, G., Moataz, F.Z.: SRLG-diverse routing with the star property. In: Proceedings of the 9th International Conference on the Design of Reliable Communication Networks, DRCN 2013, pp. 163–170. IEEE, 4–7 March 2013

Teresa Gomes received a Ph.D. in Electrical Engineering from University of Coimbra. She is Assistant Professor at the Department of Electrical Engineering and Computers of the University of Coimbra, Portugal and a researcher at the INESC Coimbra. Her main present interests are routing, protection and reliability analysis models and algorithms for optical, MPLS and GMPLS networks.

Miguel Soares received a M.Sc. degree in Electrical Engineering (Telecommunications) from University of Coimbra. This work started under during a scholarship at INESC Coimbra, and was completed while he was a software engineer at PT Inovação. Presently he is also a software engineer at Coriant. His main research interest at the moment is GMPLS.

José Craveirinha is Full Professor in Telecommunications at the Coimbra University-Portugal and coordinator of research in Network Design-INESC-Coimbra. He obtained: M.Sc.; Ph.D. in Electrical Engineering Science at the University of Essex (UK); Doctorate of Science-"Agregação" at Coimbra University. Previous status: Associate Professor at Coimbra University, R&D Engineer (Portugal Telecom); Director of INESC-Coimbra in 1994-99. Present interests: multicriteria routing and resilient routing models/algorithms.

Paulo Melo is a teacher at the Faculty of Economics of the University of Coimbra and a researcher at INESC Coimbra. He has got a PhD in Management. He has published in areas comprising decision science, services science, group decision support, information systems research and user experience evaluation.

Luísa Jorge received Ph.D. in Electrical Engineering (telecommunications and electronics) from University of Coimbra. She is a teacher at the Polytechnic Institute of Braganca and a researcher at INESC Coimbra. Her main research interests at the moment are MPLS/GMPLS, QoS analysis in multiservice telecommunications network, networking simulation and teletraffic engineering.

Vitor Mirones has a Engineering degree in Telecommunications from the University of Oporto since 1994. In PTInS since 1995 was initially integrated in the Multimedia and IP Services department. Since 2002 he was moved to the Network Systems Development department working in Network Management Solutions. He currently leads the Network Systems Management Applications. Starting in 2007 he has begun a PhD (MAPTele) on Minho, Aveiro and Porto Universities on autonomic management.

André Brízido received his degree in Electronic and Telecommunications Engineering at the University of Aveiro in 2007. He is currently working in the development of network management solutions at PT Inovação e Sistemas.