This is an Accepted Manuscript of an article published by Elsevier in Optical Switching and Networking on 2014, available online: http://dx.doi.org/10.1016/j.osn.2014.05.022.

Resilient arcs and node disjointness in diverse routing

Teresa Gomes[a,b,*], Mateusz Żotkiewicz[c]

[a]*Department of Electrical and Computer Engineering, University of Coimbra, 3030-290 Coimbra, Portugal*
[b]*INESC Coimbra, Rua Antero de Quental 199, 3000-033 Coimbra, Portugal*
[c]*Institute of Telecommunications, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warszawa, Poland*

Abstract

In multi-layer networks protection can be provided at multiple layers. Hence some links at an upper layer may be *resilient* because they are protected at a lower layer. We will designate as *resilient arc* at a given layer, an arc which has some form of protection at an underlaying layer. When path diversity is used at an upper layer, and resilient arcs are taken into account, it may not be necessary for the considered paths to be fully disjoint.

We solve a problem of finding the shortest node-disjoint pair of paths that can share resilient arcs. It is assumed that a network consists of a set of nodes and a set of arcs. Moreover, a number of available arcs are resilient. Our goal is to find the shortest pair of paths such that they share only those nodes that are incident to shared resilient arcs. Moreover, we assume that the resulting paths cannot contain loops, and costs of shared resilient arcs are counted only once towards the objective function.

In the paper we present two novel algorithms solving the above problem, and two supporting algorithms that are utilized as subroutines. We implement the proposed algorithms and compare them to an MIP approach.

Keywords: routing, protection, disjoint paths

1. Introduction and motivation

The deployment of broad-band services has increased the bandwidth demand of telecommunication networks. Satisfying this demand has been made possible by optical networks, which carry a huge amount of traffic. A single link failure in an optical network can have a significant impact in the carried traffic. Because many critical infrastructures depend on the correct and continuous operation of telecommunication networks, these networks must be resilient [2].

☆The idea was originally presented at RNDM 2013 - 5^{th} International Workshop on Reliable Networks Design and Modeling, co-located with ICUMT 2013 Congress (Sept. 10-12, 2013, Almaty, KZ) under title "Finding the shortest node-disjoint pair of paths that are allowed to share resilient arcs" [1]

*Corresponding author
Email addresses: teresa@deec.uc.pt (Teresa Gomes), mzotkiew@tele.pw.edu.pl (Mateusz Żotkiewicz)

Path diversity is a form of protection that consists in routing a traffic demand, between two specific node pairs, using a set of paths. The survivability of traffic flows, in single links (or node) failure scenarios, can be 100% guaranteed by a pair of link-disjoint (or node-disjoint) paths [3]. In multi-layer networks protection can be made at multiple layers. Hence some links at an upper layer may be *resilient* because they are protected at a lower layer. We will designate as *resilient arc* at a given layer, an arc which has some form of protection at an underlaying layer. When path diversity is used at an upper layer, and resilient arcs are taken into account, it may not be necessary for the considered paths to be fully disjoint.

In [4] the Partial Disjoint Path algorithm is proposed to avoid protection duplications in a multi-layer scenario. It is a two step approach: first it calculates the AP (active path or working path or primary path) , and then in it seeks to obtain the shortest BP (backup path or secondary path) which is disjoint with AP only in the unreliable links. The resulting paths are designated failure-disjoint paths because they do not share any risk of failure (assuming nodes do not fail) because they only share resilient arcs. Although this approach allows for sharing of resilient links (counted only once) between the AP and BP, it does not calculate, among all such path pairs, the one of minimal total (additive) cost. Hence the approach of [4] may require more network resources than would have been strictly necessary. This problem is solved in [5], where a polynomial time algorithm for finding failure-disjoint paths, of min-sum cost is proposed.

Node disjointness implies link disjointness, so using node disjoint path pairs (or sets of paths) results in higher network availability. Considering there exist resilient arcs in the network layer, where a node disjoint path pair is to be calculated, it may be advantageous from the point of bandwidth usage or cost to admit paths that share those arcs. This implies that the node disjointness restriction, at the end nodes of shared resilient arcs, would have to be relaxed. These resilient arcs may for instance represent interconnections of two sub-networks of a single operator using a third party network. Also, in a Multi-Protocol Label Switching Transport Profile (MPLS-TP) network, resilient arcs may represent (protected) tunnels at the Wavelength Division Multiplexing (WDM) layer. Real-world communication networks are generally made up of different layers, for example, a SONET/SDH layer over an optical network layer. A link of the SONET/SDH layer can be considered as a demand that should be routed through a path in the optical layer. If this path is protected against failures, then the link is reliable. Otherwise the link can fail and hence requires some protection against failures at the SONET/SDH layer. This leads to two kinds of arcs: perfectly reliable arcs that do not fail, and unreliable arcs that can fail [6, 3]. This kind of problem is consider for instance in [7], where the authors replace the optical protection with the protection in higher layers for some links. Another example are Free Space Optics (FSO) networks [8] facilitated with fiber connections for some vital routes. In this case, the technology used for fiber links makes them much more resilient to failures than the other links that are built on FSO [9].

In the context described above a new problem arises, which is the calculation of a pair of paths, from node s to node t, such that they are node-disjoint, except possibly at the end nodes of shared resilient arcs. Moreover, the returned pair of paths cannot contain loops. In this work a formal, mathematical description of the problem of calculating the shortest

node-disjoint pair of paths that are allowed to share resilient arcs, is proposed and two novel algorithms are introduced for solving it.

As for general requirements that have to be met in practice by such algorithms consider a Multi-Protocol Label Switching (MPLS) context. In the Generalized Multi-Protocol Label Switching (GMPLS) architecture a Path Computation Element (PCE) [10] is a computational unit that calculates routes as requested by a Path Computation Client (PCC). A PCE can be implemented in any system; embedded in a network element or a network management system, or implemented as a separate server dedicated to path computations [11]. The route calculation made by a PCE, can be made in a centralized or distributed manner [10]. A centralized PCE usually has good processing capabilities, and it may have a response time in the order of a few seconds, answering to requests from the network management system. In a distributed model the PCE, which may be embedded in a network element, should be able to provide a rapid response, although it may have limited calculation power and memory resources. For end-to-end protection in GMPLS networks, considering that information about resilient links is distributed, the PCE should be capable of calculating failure-disjoint path pairs, to avoid protection duplications. This shows the importance of developing efficient algorithms[1] for determining failure-disjoint paths.

The paper is organized as follows. In Section 2 we present a short literature review on resilient routing. In Section 3 we present a formal, mathematical description of the problem. This is followed by a description of related failure-disjoint problems in Section 4. Problems described there at first sight may look similar to our problem. In Section 5 supporting algorithms are presented. They are utilized as subroutines in the core algorithms solving our problem presented in Section 6. Numerical results are presented in Section 7. The paper ends with conclusions in Section 8.

2. Literature review of resilient routing issues

In [12] networks challenges are defined as adverse events causing faults, that may result in service failures. Network service providers seek to ensure their networks' survivability, so that when a fault occurs its impact is strongly mitigated and in many cases not even perceived by the users. To attain this objective, two main recovery options are available: protection and restoration. In the first case, backup resources are reserved in advance, before any fault occurs. In the second case, after detecting a fault a solution to recover the affected connection is searched for, and the necessary resources are then allocated to restore the affected services. Protection has higher cost, but strict recovery time, and is the preferred solution at the optical layer [13]. Restoration is more bandwidth efficient than protection, as it makes better use of network resources. However, the efficiency is at the expense of longer recovery times, and hence should only be used for services which can tolerate some disruption resulting in worse quality of service.

Regarding the scope, the recovery can be global, local, or segment oriented [6]. When global path protection is utilized, the AP is utilized in normal network operation conditions.

[1]This work was in fact motivated by a R&D Project under contract with PT Inovação.

When a fault occurs (due to a node or link failure) a fault indication signal (FIS) is generated by the node closest to the failed element and is sent towards the head end of the path. Once the head end of the failed AP receives the FIS, it will switch the traffic to the BP. When local protection is used the node closest to the fault switches the traffic to a pre-established detour, which avoids the failed link or node. This results in a faster recovery than global protection, but requires much more backup resources. In segment protection the AP is divided into segments, which are locally protected; segment protection is a compromise solution: it is faster that global protection and requires less backup resources than local protection [6].

Dedicated protection ensures fast recovery but requires a high level of network redundancy (ratio of backup to active bandwidth capacity). In a single failure scenario protection resources can be shared among BPs of risk disjoint APs. When shared protection is considered the BPs are not established (together with APs) but are pre-calculated, thus allowing them to share protection bandwidth. This implies a longer restoration time, but results in significant reduction of network redundancy [14].

When dedicated protection is used, a link-disjoint (or node-disjoint) path pair can be obtained in polynomial time [15]. In shared path protection, the calculation of an adequate AP and BP can be modeled as min-sum problem with ordered dual costs, which has been shown to be NP-Complete [16, 17]. Algorithms for solving the min-sum problem with asymmetric weights can be found in [18] and [19]. A more general problem, the min-sum with dual costs, is solved in [20]. The algorithms in [18, 19, 20] are all variants of the Iterative Two Step Approach, which consists in using a k-shortest enumeration algorithm [21, 22], for successively obtaining a candidate active path for which an adequate backup path is calculated. The iterative process continues until certain predefined criteria are met.

The problem of calculating an AP and a BP considering shared bandwidth is tackled in [14]. An Integer Linear Programming (ILP) approach is proposed there, but because the calculation of the AP and BP are entwined, solving exactly the least-cost AP and BP path-pair (in terms of the sum of the cost) is not scalable in the network size. Hence in [14] two heuristic algorithms, designated ITSA and Maximum Likelihood Relaxation (MLR), are proposed. The authors conclude that ITSA, although giving the best results, can be computationally expensive and that MLR is a good compromise between performance and computational efficiency.

In [23] the approach called the designated PRotection using Multiple SEgments (PROMISE) for shared segment protection is proposed. It seeks, for a given node pair, the best AP segments (ASs) and corresponding BP segments (BSs), considering the ASs and BSs may overlap. In [23] an ILP formulation and an heuristic for solving the problem of calculating the BSs for a given AP are presented.

Fast ReRouting (FRR) [24] in MPLS networks implements local protection considering bandwidth sharing among the BPs of each protected AP (intra-demand sharing). When inter-demand bandwidth sharing is desired, the calculation of the AP an BP (and the capacity for bandwidth sharing) depends on the considered information model. In [25, 26] three scenarios are considered: no information (NI), aggregated information (AG), and complete information (CI). The NI model allows for intra-demand sharing, but no inter-demand sharing is possible, because only the residual bandwidth in each link is known to the routing

algorithms. The CI model could be used with centralized routing, but is not realistic in the context of on-line (distributed) routing, due the huge amount of information that would have to be distributed by the routing protocols (and maintained at each node). Finally the AG model, where aggregated information about the bandwidth reserved for APs and BPs, and the residual bandwidth in each link is distributed, allows for inter-demand sharing. Distributed Partial Information Management schemes that ensure sufficient BP bandwidth estimation using the AG model, but allow minimum BP bandwidth allocation (and deallocation), are proposed in [27], but require some extensions of the signaling protocols.

In [28] a generic optimization framework, referred to as the Generalised Shared Protection (GSP), is proposed. It selects the AP and a type (link, segment, or global) of shared protection for a certain end-to-end demand under given traffic and network conditions. GSP, using an ILP formulation, calculates the BP (path, segments, or detours) which uses resources optimally along with the AP for any single demand.

The preconfigured protection cycles (p-cycles) proposed by Grover [29] are characterized by a recovery speed similar to SDH/SONET rings and present mesh-like efficiency because p-cycles can protect both on-cycle and straddling links. A review of several p-cycle based protection (for link, segment, path, and flow) approaches can be found in [30]. p-cycles can also be utilized to protect multicast traffic trees [30, 31].

IP-over-WDM (Wavelength-Division Multiplexing), is one of the considered architectures for the next generation Internet. Survivability is of the utmost importance in this type of networks, because of the huge amount of traffic carried in a single light-path. The most common approach is to consider protection at the WDM layer or in alternative at the IP layer [13]. Depending on a failure scenario and a traffic pattern it may be more advantageous (considering backup capacity) to ensure protection at the optical layer in some cases, and in other cases to ensure recovery at the IP layer. In [13] the bandwidth requirements of IP restoration versus optical protection for a given class of services is analyzed and a combination of both recovery types is also considered. Zhang *et al.* [32] study how to maximize the bandwidth sharing for survivable IP/MPLS over WDM, and propose a new scheme for resource sharing, which is more effective than two other alternatives.

The GMPLS control plane [33] has been standardized by the Internet Engineering Task Force (IETF). Recovery in GMPLS networks has also been specified in IETF RFC 4872 [34]. The Resource Reservation Protocol-Traffic Engineering (RSVP-TE) with extensions [35] is used for signaling the restoration path according to the calculated route of the backup path. Sharing the bandwidth in the common link(s) of the path pair (active LSP and backup LSP) can be easily implemented in the MPLS control plane. It suffices that they are signaled using the Shared-Explicit (SE) style of RSVP-TE [36]. This allows to take advantage of the existence of resilient links, and avoids wasting backup bandwidth in a multi-layer scenario.

3. Problem Formulation

In this section we present a problem of obtaining the min-sum path pair, such that the paths are node-disjoint, except possibly at the end nodes of shared resilient arcs. Moreover, the paths cannot contain cycles. In order to formally present the optimization problem we

introduce the following notation.

Sets:

$\mathcal{V}$ set of nodes; a single node is denoted by v

$\mathcal{A}$ set of arcs; a single arc is denoted by a; each arc is an ordered pair (v, v') of elements from $\mathcal{V}$

$\mathcal{S}$ set of resilient arcs; $\mathcal{S} \subseteq \mathcal{A}$

$\delta^+(v)$ set of arcs emergent from node v

$\delta^-(v)$ set of arcs incident in node v

Parameters:

c_a cost of arc a; $c_a > 0$ for all $a \in \mathcal{A}$

s source node; $s \in \mathcal{V}$

t target node; $t \in \mathcal{V}$

Variables:

x_a number of paths using arc a

r_a 1 if arc a is used in its resilient mode; 0 otherwise

r_v 1 if node v is used in its resilient mode; 0 otherwise

An arc is used in its resilient mode if it is being shared by the AP and BP. A node is used in its resilient mode if it is the end node of a shared resilient arc.

The considered optimization problem—Shortest Node-Disjoint Pair of Paths Sharing Resilient Arcs—will be denoted by SNDP^{2}SRA, and can be formulated as follows.

$$\min \quad \sum_{a \in \mathcal{A}} c_a (x_a - r_a)$$

$$2 \cdot r_a \leq x_a \quad \forall a \in \mathcal{A} \tag{1a}$$

$$\sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = \begin{cases} 2 & \text{if } s = v \\ -2 & \text{if } t = v \\ 0 & \text{otherwise} \end{cases} \quad \forall v \in \mathcal{V} \tag{1b}$$

$$x_a \leq 1 \quad \forall a \in \mathcal{A} \setminus \mathcal{S} \tag{1c}$$

$$r_v \leq \sum_{a \in \delta^+(v) \cup \delta^-(v)} r_a \quad \forall v \in \mathcal{V} \tag{1d}$$

$$\sum_{a \in \delta^+(v)} x_a \leq 1 + r_v \quad \forall v \in \mathcal{V} \setminus \{s\} \tag{1e}$$

$$x_a \in \{0, 1, 2\} \quad \forall a \in \mathcal{A} \tag{1f}$$

$$r_a \in \{0, 1\} \quad \forall a \in \mathcal{A} \tag{1g}$$

$$r_v \in \{0, 1\} \quad \forall v \in \mathcal{V} \tag{1h}$$

In (1) we pay for each utilized arc. However, if an arc is used by both paths ($x_a = 2$), then this arc should be counted only once towards the objective function. This fact is controlled by r_a, which is bounded by (1a). Equation (1b) expresses the standard flow conservation law for variables x_a. Variables x_a are additionally bounded by inequality (1c), which ensures that unreliable arcs cannot be shared by the paths. A node is used in its resilient mode only if at least one incident arc is also used in its resilient mode. This fact is expressed by (1d). Nodes not used in their resilient modes cannot be shared by the paths. This restriction is represented by (1e). Inequality (1e) assures that if a node is not resilient, it cannot be left by more than one path (except source node s). What is more, inequality (1e) also assures that loops will not be present in the solution.

4. Related min-cost failure-disjoint problems

Our problem can be easily mistaken for another failure-disjoint problems that can be found in the literature. In this section we want to briefly describe a number of key features that distinguish our problem from the other similar problems. Let us summarize them now.

1. A solution to our problem is a pair of node-disjoint paths.
2. The pair can share "resilient" nodes.
3. A node is resilient only if it is adjacent to a resilient arc.
4. Both paths of the pair have to use this resilient arc to make the adjacent nodes resilient.
5. The paths cannot contain loops.
6. If both paths are using the same resilient arc, we pay for it only once.

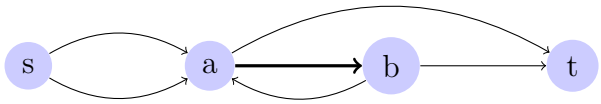

Figure 1: Network showing the importance of Feature 5.

If a problem satisfies only Feature 1, then it can be easily solved by Suurballe's algorithm [15]. This algorithm can be also used when a solution has to be additionally characterized by Features 2 and 3. However, when Feature 4 has to be also satisfied, then a new specialized algorithm is needed. Here it is worth to notice that a problem, which lacks Feature 5, still differs from the problem considered in this work. It is clearly visible in Fig. 1. Consider a problem of finding a pair of failure-disjoint paths from s to t in the network of Fig. 1 with one resilient link depicted with a thick arc. In such a network there is a feasible solution satisfying Features 1, 2, 3, and 4. However, there is no solution when also Feature 5 has to be satisfied.

Finally, Feature 6 deals with costs and can be used independently of the others. The feature is very important as it can change the way problems can be solved. Consider a problem characterized by Features 1, 2, and 3. As written before, it can be solved by Suurballe's algorithm. However, when Feature 6 is added, then a new specialized algorithm has to be presented. The same situation is with our problem. Algorithms presented in this paper cannot solve a problem that is characterize by a cost function which does not satisfy Feature 6.

5. Supporting algorithms

In this section three supporting procedures are presented. They are used as subroutines in our main algorithms presented in the next section. The procedures use a notion of a path instead of a flow on arcs like in (1). The paths are denoted by p and q, and a pair they form is denoted by (p, q). The cost of the pair is denoted by $c(p, q)$:

$$c(p,q) = \sum_{a \in p} c_a + \sum_{a \in q} c_a - \sum_{a \in q \wedge a \in q} c_a \tag{2}$$

Notice that the common arcs in p and q must belong to $\mathcal{S}$.

5.1. Failure-Disjoint Path Algorithm [5] (FDP)

Before moving to actual supporting algorithms let us first briefly describe an algorithm presented in [5] that solves a problem of finding a min-sum path pair that can share resilient arcs paying for them only once. The procedure is presented in Algorithm 1. We will refer to it as Failure-Disjoint Paths Algorithm (FDP). The algorithm is used as a subroutine in the first supporting procedure, while the second supporting procedure is in fact a modified version of FDP. As proven in [5] the algorithm is polynomial.

Algorithm 1 Failure-Disjoint Paths Algorithm (FDP)

Require: Digraph $G = (\mathcal{V}, \mathcal{A})$; source node s; target node t; arc costs c_a such that $a \in \mathcal{A}$; set of unreliable arcs $\bar{\mathcal{S}} = \mathcal{A} \backslash \mathcal{S}$.

Ensure: (p, q) is the min-sum cost failure-disjoint path pair from s to t, or $(\emptyset, \emptyset)$ if no solution was found.

1: Compute the shortest pair of arc-disjoint paths from each vertex u (sources) to each vertex v (sinks), where u is either s or a sink of any resilient arc, and v is either t or a source of any resilient arc.

2: In the resulting modified graph, which consists of pairs of arc-disjoint paths (represented by newly added arcs) and of resilient arcs, compute the shortest path.

3: Consider a graph consisting of doubled resilient arcs selected by the shortest path algorithm of Step 2 and arcs that form pairs of arc-disjoint paths computed in Step 1 and also selected by the same run of the shortest path algorithm (Step 2). Compute a pair (p, q) of arc-disjoint paths in the considered graph.

4: **return** path pair (p, q) found in the previous step or $(\emptyset, \emptyset)$ if no solution was found.

5.2. Failure-Disjoint Paths Algorithm considering Node Failures (FDP-NF)

The first approach uses FDP for obtaining a failure-disjoint path pair of min-sum cost in an appropriately modified graph. The approach is presented in Algorithm 2. We will refer to it as Failure-Disjoint Paths Algorithm considering Node Failures (FDP-NF).

FDP finds a min-sum cost failure-disjoint path pair in networks with a number of reliable arcs, and uses the set of sink nodes (t and the sources of resilient arcs) and the set of source nodes (s and the sinks of resilient arcs) to achieve this goal. However, in our problem a node-disjoint pairs of paths are of interest. Therefore, we will assume that the nodes not in the set of sinks or in the set of sources are now *unreliable nodes*, and in order to cope with unreliable nodes, before running FDP, a network transformation, as already suggested in [5], will be made, where every unreliable node is divided in two sub-nodes connected by an unreliable arc of zero cost.

More precisely, let v be an unreliable node divided into two sub-nodes, v_1 and v_2, connected by an unreliable arc (v_1, v_2) of zero cost; every incident arc in node v will now be incident in node v_1 and every emergent arc from v will now emerge from v_2. This will ensure that the failure-disjoint (in fact arc-disjoint in the unreliable arcs) paths calculated in line 1 of FDP-NF are also node-disjoint in the considered unreliable nodes.

In line 2 algorithm FDP-NF calculates the shortest pair of failure-disjoint paths. This path pair does not share any unreliable arcs, and hence must be node-disjoint on the unreliable nodes (the nodes not in the set of sinks or sources), due to the network transformation in line 1. In line 3 the graph transformation of line 1 is reversed to return a path pair in the original graph G.

Theorem 1. *A path pair returned by FDP-NF may only share resilient arcs and the considered resilient nodes, and is of min-sum cost among such path pairs.*

Algorithm 2 Failure-Disjoint Paths Algorithm considering Node Failures (FDP-NF)

Require: Digraph $G = (\mathcal{V}, \mathcal{A})$; source node s; target node t; arc costs c_a such that $a \in \mathcal{A}$; set of unreliable arcs $\bar{\mathcal{S}} = \mathcal{A} \backslash \mathcal{S}$.

Ensure: (p, q) is the min-sum cost failure-disjoint path pair from s to t which is node-disjoint except possibly at end nodes of resilient arcs, or $(\emptyset, \emptyset)$ if no solution was found.

1: Transform the graph, replacing every unreliable node by two nodes and an unreliable arc.

2: Use FDP to find the shortest pair of failure-disjoint paths from s to t in the transformed graph.

3: If a path pair was found in the previous step, collapse every divided node in the paths into the original node.

4: **return** path pair (p, q) found in the previous step or $(\emptyset, \emptyset)$ if no solution was found.

PROOF. The network transformation of line 1 replaces the considered unreliable nodes (the nodes which are not s, t or the end nodes of resilient arcs) by unreliable arcs. The result of calling FDP in line 2 of FDP-NF, whose correctness is proved in [5], is the min-sum failure-disjoint path pair in that transformed graph. Hence the resulting path pair may only share resilient arcs and the considered resilient nodes, and must be of min-sum cost among such path pairs.

Therefore, the path pair resulting from FDP-NF may share resilient arcs, will be node disjoint except possibly at the end nodes of resilient arcs and will have min-sum cost. However, have in mind that FDP-NF does not ensure that the shared nodes are end nodes of shared resilient arcs. In fact, the resilient arcs do not have to be used at all.

Example FDP-NF. Consider a network presented in Fig. 2 with a resilient arc depicted using a thick arrow and all arc costs like in the figure. Assume that a pair of paths from s to t is searched for. After applying the network transformation of line 1 of FDP-NF, we obtain a network presented in Fig. 3. Notice that only nodes b and d are not either a source, a sink, or are incident to a reliable arc. Therefore, only those nodes are replaced with two nodes b_1 and b_2 (d_1 and d_2, respectively), and an unreliable arc between them. Obviously arcs that were originally incident in b or d are now incident in b_1 or d_1, while arcs that were emergent from b and d are now emergent from b_2 and d_2.

In line 2 of FDP-NF the shortest failure-disjoint path pair is found in the transformed graph. In our example there are only two such pairs (of cost 7), and they both use all arcs in the graph except of (b, a). Consider a pair $s \to b_1 \to b_2 \to c \to t$ and $s \to a \to c \to d_1 \to d_2 \to t$ (the second pair is $s \to a \to c \to t$ and $s \to b_1 \to b_2 \to c \to d_1 \to d_2 \to t$). After reducing the graph to its original form in line 3 of FDP-NF the paths will reduce to $s \to b \to c \to t$ and $s \to a \to c \to d \to t$ (or to $s \to a \to c \to t$ and $s \to b \to c \to d \to t$ for the second pair). Obviously those are not feasible solutions to SNDP^{2}SRA, as arc (a, c) is not shared by the paths, thus also neither a nor c can be shared by the paths—in both obtained solutions c is shared by the pairs. As clearly seen in the example, FDP-NF cannot solve SNDP^{2}SRA.

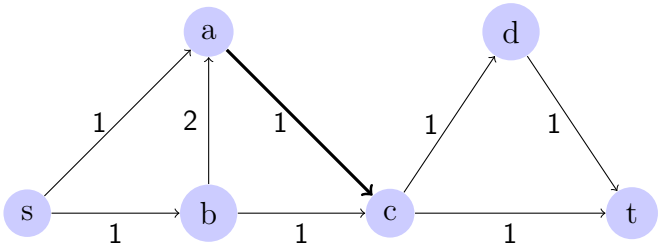

Figure 2: Example network.

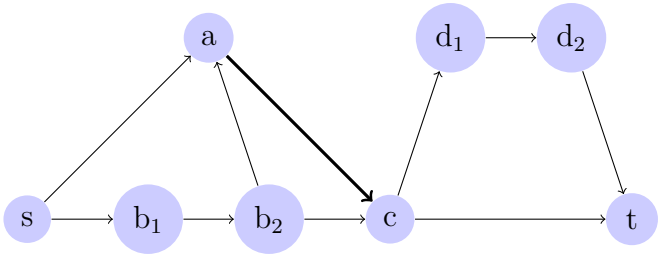

Figure 3: Transformed network.

5.3. Advanced Failure-Disjoint Paths Algorithm considering Node Failures (AdvFDP-NF)

Another approach is to use a variant of FDP, for obtaining a failure-disjoint path pair of min-sum cost. The approach is more efficient than FDP when the percentage of resilient edges is relatively small. However, it loses its competitive edge when the percentage exceeds approximately 50%. The approach will be called Advanced Failure-Disjoint Paths Algorithm considering Node Failures (AdvFDP-NF) in the rest of the paper.

AdvFDP-NF is essentially Algorithm 1 except for:

- calculating node-disjoint path pairs in line 1 instead of arc-disjoint path pairs;
- obtaining a maximally node-disjoint path pair [37] in line 3, in order to allow sharing the resilient arcs (and corresponding end nodes) instead of finding two arc-disjoint paths in the reduced graph with resilient arcs.

The algorithm is not as universal as FDP-NF, because the problem it solves makes sense only in the context of SNDP^{2}SRA. FDP-NF returns the min-sum cost failure-disjoint path pair from s to t which is node-disjoint except possibly at end nodes of resilient arcs. On the other hand, certain things that we can write about results of AdvFDP-NF are much less general. Consider an artificial problem AdvFDP-NF-Problem that is solved by AdvFDP-NF. We can write that:

Lemma 1. *All feasible solutions to SNDP^{2}SRA are also feasible to AdvFDP-NF-Problem,* i.e.*, if a feasible solution to SNDP^{2}SRA exists in $\mathcal{G}$, then also AdvFDP-NF returns a feasible solution in $\mathcal{G}$.*

PROOF. First notice that if a solution is feasible for AdvFDP-NF-Problem in a given graph $\mathcal{G}$, then it will be also feasible in any graph that contains $\mathcal{G}$, as we can always set c_a for all additional arcs to infinity. The reason is that c_a values do not have an impact on the feasibility of a solution. Still, arcs with c_a equal to infinity will not be in an optimal solution as long as a feasible solution not containing such arcs exists.

Consider now any feasible solution to SNDP^{2}SRA in a given graph $\mathcal{G}$. Remove from $\mathcal{G}$ all the arcs that are not in the considered feasible solution. Notice that if AdvFDP-NF finds the same solution as the considered feasible solution in this reduced graph, then all feasible solutions to SNDP^{2}SRA are also feasible for AdvFDP-NF-Problem. A feasible solution to SNDP^{2}SRA consists of segments that are either a shared resilient arc or a node-disjoint pair of paths. The segments are ordered in such a way that no two segments that are formed by a pair of node-disjoint paths are adjacent to each other. In a graph consisting solely of arcs that are in a feasible solution to SNDP^{2}SRA after running line 1 of Algorithm 3 (AdvFDP-NF) we obtain a chain, which will be a modified graph in line 2. It leads us to a solution that contains exactly the same arcs that were used by a solution to SNDP^{2}SRA, which ends the proof.

Lemma 2. *Objective values of the same solutions for AdvFDP-NF-Problem and SNDP^{2}SRA are equal.*

PROOF. In SNDP^{2}SRA a cost of a solution is a sum of c_a of each arc belonging to a solution. A cost of a solution to AdvFDP-NF-Problem consists of a sum of costs of node-disjoint pair of paths and a sum of costs of used resilient arcs, which is in fact a sum of c_a of each arc belonging to a solution.

Lemma 3. *All feasible solutions to AdvFDP-NF-Problem are also feasible to FDP-NF-Problem,* i.e.*, if AdvFDP-NF returns a feasible solution in $\mathcal{G}$, then also FDP-NF returns a feasible solution in $\mathcal{G}$.*

PROOF. Consider the lemma is not true, and there exists a feasible solution to AdvFDP-NF-Problem that is not feasible to FDP-NF-Problem. An infeasible solution to FDP-NF-Problem either contains loops or two paths of the solution share a node which is not adjacent to a reliable arc. Notice that a solution to AdvFDP-NF-Problem cannot contain loops as $c_a > 0$ for all $a \in \mathcal{A}$, so any loop would be eliminated by the maximally min-sum node-disjoint paths algorithm of line 3 of AdvFDP-NF. Therefore, consider that there exists a solution to AdvFDP-NF-Problem with two paths sharing a node which is not adjacent to a reliable arc. Such a solution can be obtained only if the shared node alone creates in the reduced graph a vertex cut. However, if the node is not adjacent to any resilient arc, so the reason it was included in the reduced graph was because it was on one of node-disjoint pairs of paths computed in line 1 of AdvFDP-NF. Therefore, it cannot form alone a vertex cut in the reduced graph. Thus, a contradiction is reached.

Lemma 4. *Objective values of the same solutions for AdvFDP-NF-Problem and FDP-NF-Problemare equal.*

PROOF. A cost of a solution to FDP-NF-Problem is a sum of costs of used arcs. As shown in the proof of Lemma 2, a cost of a solution to AdvFDP-NF-Problem is also a sum of costs of all used arcs.

Let us now draw some important conclusions from the lemmas. First of all, the first two lemmas tell us that AdvFDP-NF-Problem, like FDP-NF-Problem, are relaxations of SNDP^{2}SRA. It will be important from the point of view of algorithms presented in Section 6, as only procedures solving relaxations of SNDP^{2}SRA can be used in those algorithms as subroutines.

On the other hand, the following two lemmas (3 and 4) tell us that FDP-NF-Problem is a relaxation of AdvFDP-NF-Problem. This fact makes AdvFDP-NF not worse than FDP-NF as a subroutine of the algorithms of Section 6. In fact, as show in numerical results of Section 7, the fact makes AdvFDP-NF slightly more efficient than FDP-NF while used as the subroutine.

Algorithm 3 Advanced Failure-Disjoint Paths Algorithm considering Node Failures (AdvFDP-NF)

Require: Digraph $G = (\mathcal{V}, \mathcal{A})$; source node s; target node t; arc costs c_a such that $a \in \mathcal{A}$; set of unreliable arcs $\bar{\mathcal{S}} = \mathcal{A} \backslash \mathcal{S}$.

Ensure: (p, q) satisfies lemmas 1, 2, 3, and 4, or $(\emptyset, \emptyset)$ if no solution was found.

1: Find 2 min-sum node-disjoint paths from sources to sinks.
2: Find the shortest path in the modified graph, which consists of pairs of node-disjoint paths (represented by newly added arcs) and of resilient arcs.
3: Find 2 maximally min-sum node-disjoint paths in the reduced graph, if the shortest path in that graph is not entirely reliable.
4: **return** path pair (p, q) found in the previous step or $(\emptyset, \emptyset)$ if no solution was found.

AdvFDP-NF is formally presented in Algorithm 3. In line 1 it calculates the shortest pair of node-disjoint paths from each node u (sources) to each node v (sinks), where u is either s or a sink of any resilient arc, and v is either t or a source of any resilient arc. In line 2 the modified graph corresponds to the original graph, where all the unreliable arcs have been removed and for each calculated node-disjoint path pair one arc from the source of the pair to the sink of the pair is added to the graph with the cost of the corresponding path pair. The reduced graph of line 3 is the graph induced by the the resilient arcs selected by the shortest path algorithm in line 2 and the set of arcs that form pairs of node-disjoint paths computed in line 1 and also selected by the same run of the shortest path in line 2. Calculating two maximally node-disjoint paths [37] in the reduced graph allows the sharing of resilient arcs and its end nodes.

The path found in line 2 of AdvFDP-NF is formed by zero or more resilient arcs and/or by arcs which represent 2 node-disjoint paths from sources to sinks obtained in line 1 of AdvFDP-NF. However, the arcs belonging to the path obtained in line 2 can create a solution with cycles. Therefore, 2 maximally node-disjoint paths have to be found in the reduced graph as stated in line 3.

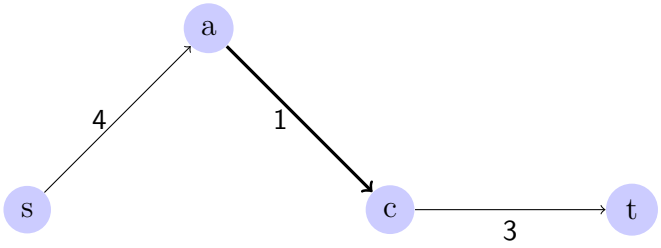

Figure 4: Modified network for AdvFDP-NF.

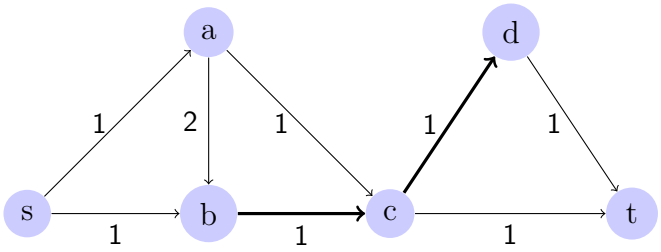

Figure 5: Example network for AdvFDP-NF.

Example AdvFDP-NF. Again consider a network presented in Fig. 2 with the same arc costs and starting and ending nodes. After applying the network transformation of line 2 we obtain a network presented in Fig. 4. In this network there is only one path from s to t (of cost 8). After extending the graph to its original form in line 3 of AdvFDP-NF the path will extend to a pair $s \to b \to a \to c \to t$ and $s \to a \to c \to d \to t$ (or to $s \to a \to c \to t$ and $s \to b \to a \to c \to d \to t$ for the second pair).

Those are feasible solutions to SNDP^{2}SRA. However, AdvFDP-NF alone is still not able to solve SNDP^{2}SRA. Consider a slightly different network of Fig. 5 with two reliable arcs, and arc (b, a) replaced with (a, b). In this network an optimal solution to AdvFDP-NF would cost 7, while an optimal solution to SNDP^{2}SRA would cost 8.

6. Algorithms solving SNDP^{2}SRA

In this section two algorithms solving SNDP^{2}SRA are presented. They both use the supporting algorithms of Section 5 (FDP-NF and AdvFDP-NF) as subroutines.

Before moving to the description of the algorithm, let us introduce some additional notation. Let v be the first (from source to target) intermediate common node in a path pair obtained using a supporting algorithm, which is not the end node of a shared resilient arc:

- $\phi^+(v)$ pair of arcs emergent from v which are *present in the path pair* obtained using a supporting algorithm.
- $\phi^-(v)$ pair of arcs incident on v which are *present in the path pair* obtained using a supporting algorithm.

- $\delta^+_{\mathcal{S}}(v) = \delta^+(v) \cap \mathcal{S}$, is the set of resilient arcs emergent from v.

In our example for FDP-NF the first intermediate common node in the path pair obtained using the algorithm, which is not the end node of a shared resilient arc, is c. Therefore, sets $\phi^+(c)$ and $\phi^-(c)$ are equal to $\{(c,d),(c,t)\}$ and $\{(b,c),(a,c)\}$, respectively. Finally, $\delta^+_{\mathcal{S}}(c) = \emptyset$.

6.1. Simple approach

In this section a basic solution to SNDP^{2}SRA is presented. In order to continue, let us prove a basic lemma that will be the first step to present the algorithm.

Lemma 5. *If a supporting algorithm returns an admissible solution to SNDP^{2}SRA, then this is the optimal solution to SNDP^{2}SRA.*

Proof. First notice that each solution to SNDP^{2}SRA is also an admissible solution to AdvFDP-NF (Lemma 1) and FDP-NF (Lemma 1 combined with Lemma 3). Moreover, objective functions of those three problems are identical (Lemmas 4 and 2), *i.e.*, the same pair of paths will have the same cost for SNDP^{2}SRA, AdvFDP-NF, and FDP-NF. If the returned solution was not the optimal solution to SNDP^{2}SRA, then there would exist another solution to SNDP^{2}SRA with a lower cost. However, this another solution would be also an admissible solution to both FDP-NF and AdvFDP-NF. Therefore, our solution not only could not be an optimal solution to SNDP^{2}SRA but also could be an optimal solution neither to FDP-NF nor to AdvFDP-NF, thus a contradiction is reached.

When the returned solution is not a feasible solution to SNDP^{2}SRA, then a simple approach to fix it consists in a successive removal of each of the arcs (incident and emergent) in the first common intermediate node (which is not the end node of a shared resilient arc) from source to target, and then applying a supporting algorithm to every one of these pruned networks, which will be designated as the new problems to solve. Let us consider only FDP-NF as a supporting algorithm in our explanations and examples, as using AdvFDP-NF is similar in general.

In the algorithm that follows the function *firstCNode*(p,q) returns the first (from s to t) common intermediate node between p and q, which is not the end node of a *shared* resilient arc. An admissible solution of SNDP^{2}SRA (line 12 of the next algorithm) is a node-disjoint path pair where the only intermediate nodes in common are end nodes of *shared* resilient arcs.

Let us now present the algorithm. An example of using it is given in the end of this section.

The idea is to make a Breadth First Search (BFS) in a 4-ary tree of network graphs, G' in line 9 of ASA. This ensures an exhaustive search among all networks G' that may result in a solution to the problem of finding a node-disjoint path pair except possibly at end nodes of *shared* resilient arcs; if no admissible solution is obtained, the problem has no solution.

Theorem 2. *ASA finds a feasible and optimal solution to SNDP^{2}SRA.*

Algorithm 4 Algorithm Simple Approach (ASA)

Require: Digraph $G = (\mathcal{V}, \mathcal{A})$; source node s; target node t; arc costs c_a such that $a \in \mathcal{A}$; set of unreliable arcs $\bar{\mathcal{S}} = \mathcal{A} \backslash \mathcal{S}$.

Ensure: (p, q) is the min-sum cost node-disjoint path pair from s to t except possibly at end nodes of *shared* resilient arcs, or $(\emptyset, \emptyset)$ if no solution was found.

Let $\mathcal{Q}$ be a queue of sets of arcs
Push $\emptyset$ into $\mathcal{Q}$ // Initial network is G
$(p, q) \leftarrow (\emptyset, \emptyset)$ // Initially: no solution and its cost is ∞
while $\mathcal{Q}$ is not empty **do**
    $\hat{\mathcal{A}} \leftarrow$ head of $\mathcal{Q}$
    Remove head of $\mathcal{Q}$
    $\mathcal{A}' \leftarrow \mathcal{A} \backslash \hat{\mathcal{A}}$ // Set of arcs of current problem (c.p.)
    $\bar{\mathcal{S}}' \leftarrow \mathcal{A}' \cap \bar{\mathcal{S}}$ // Set of unreliable arcs of c.p.
    $(p', q') \leftarrow$ **FDP-NF**$(s, t, G'(\mathcal{V}, \mathcal{A}'), c, \bar{\mathcal{S}}')$ // or alternatively **AdvFDP-NF**
    if $(p', q') \neq (\emptyset, \emptyset)$ **then**
        if $c(p', q') < c(p, q)$ **then**
            if (p', q') is an admissible solution **then**
                $(p, q) \leftarrow (p', q')$ // updates (p, q)
            else //New problems
                $v \leftarrow firstCNode(p', q')$
                for all $a \in \phi^+(v) \cup \phi^-(v)$ **do**
                    Push $\hat{\mathcal{A}} \cup \{a\}$ into $\mathcal{Q}$
                end for
            end if
        end if
    end if
end while
return path pair (p, q) // No solution if $(p, q) = (\emptyset, \emptyset)$

PROOF. Assume it is not true, and all optimal solutions cannot be found by the algorithm. Consider one such a solution (call it $(p,q)^\star$), and any graph G' in $\mathcal{Q}$ for which $(p,q)^\star$ is admissible. In the graph G' a solution to FDP-NF is found in line 9. The solution is not $(p,q)^\star$, as it would mean that the optimum can be found, due to Lemma 5. Its cost is less or equal $c(p,q)^\star$, as in other case $(p,q)^\star$ would not be admissible in G'. Finally, the solution is not admissible, as in other cases the solution would be optimal (and would be found). The solution is not admissible, so it produced four new problems and four new graphs G'. At least in one of them $(p,q)^\star$ is admissible, as in other case it would mean that $(p,q)^\star$ contains all four arcs from $\phi^+(v) \cup \phi^-(v)$, thus $(p,q)^\star$ is not admissible. We have proved that if $(p,q)^\star$ is admissible in G' from $\mathcal{Q}$, then it will either be found or be admissible in at least one child of G' that will be inserted in $\mathcal{Q}$. Knowing that $(p,q)^\star$ is admissible in original graph G, we conclude that it either has to be found or there will always exist at least one G' in $\mathcal{Q}$ where $(p,q)^\star$ is admissible. The graph G has a limited number of arcs, so graphs G' cannot be created infinitely, thus if an optimal solution exists, then it has to be found. The contradiction is reached.

Example ASA. Consider again the network of Fig. 2. The first execution of FDP-NF in line 9 of ASA returns one of the two solutions discussed in the previous example, *e.g.*, $s \to b \to c \to t$ and $s \to a \to c \to d \to t$. The solution exists, its cost is smaller than ∞. However, it is not admissible. Therefore, we are in line 14 of ASA. The first common node of the two paths is c. The set of arcs incident or emerging from c and on one of the paths $(\phi^+(c) \cup \phi^-(c))$ is $\{(a,c),(b,c),(c,d),(c,t)\}$. It means that four new problems are added to $\mathcal{Q}$. In the following four executions of the main loop of line 4 of ASA algorithm FDP-NF will only find a solutions when $\hat{\mathcal{A}} = \{(b,c)\}$. In the remaining three loops a solution will not be found, the loops will end at line 10, and will not add any new object to $\mathcal{Q}$.

There are two optimal solutions when $\hat{\mathcal{A}} = \{(a,c)\}$. They use exactly the same arcs, so we will concentrate only on one of them: $s \to b \to c \to t$ and $s \to a \to c \to d \to t$. The solution is admissible, so it does not add any new objects to $\mathcal{Q}$. Instead, it will be set as the optimal solution in line 13 of ASA. Finally, the algorithm will end returning this solution, after executing the main loop for the fifth time.

6.2. Efficient approach

Depending on the fact that the first (from source to target) intermediate common node v in the path pair (p',q'), which is not the end node of a shared resilient arc, obtained using FDP-NF, should be (or not) shared by the solution, there are two cases:

1. The incident arcs in v, $\phi^-(v)$, are both part of the optimal solution. Hence v must be the tail node of a resilient shared arc of the optimal solution.
   To obtain this type of a solution one must remove from the graph *all arcs incident* on v except $\phi^-(v)$, and *all unreliable arcs emergent from* v. Let this set be $\chi(v)$:

$$\chi(v) = \underbrace{[\delta^-(v) \backslash \phi^-(v)]}_{\text{incident}} \cup \underbrace{[\delta^+(v) \backslash {\delta_{\mathcal{S}}}^+(v)]}_{\substack{\text{unreliable} \\ \text{emergent}}}$$

In this case v is considered to be the tail node of a resilient shared arc of the optimal solution, so v should only belong to the sink node set. Hence node v must be removed from the set of source nodes of the new problem to be solved using FDP-NF. This can be accomplished by removing from set $\mathcal{S}'$ (where $\mathcal{S}' = A' \backslash \bar{S}'$) the resilient arcs in $\phi^-(v)$ (recall that all other incident arcs in v will be removed from $\mathcal{A}'$).
Additionally if both arcs emergent from v in the path are resilient, *i.e.*, $\delta^+_{\mathcal{S}}(v) \supseteq \phi^+(v)$, only one of them will belong to the final solution, so each of them must be pruned from the network (while the other remains) generating two new problems.

2. The incidents arcs in v, $\phi^-(v)$ cannot both be a part of the optimal solution.
To obtain this type of solution each of the incident arcs in v, $\phi^-(v)$, must be removed (one at a time), generating two new problems.

This resolution method generates new problems pruning arcs from the graph and making resilient arcs unreliable (by removing them from $\mathcal{S}$). To record these two actions, which result in new problems, a queue of pairs of sets of arcs is used. The first element of the pair is the set of pruned network arcs; the second element is the set of now unreliable arcs.

Let us now present the algorithm. As in the case of ASA, a simple example of using the algorithm is given in the end of the section.

The idea of AEA (as in ASA) is to make a BFS search in a 4-ary tree of network graphs, G' in line 9 of AEA. Therefore, a proof of correctness of the algorithm can also be based on the fact that an optimal solution admissible in G' will be either found or be admissible at least in one child of G'. This can be easily deduced from the description of the algorithm.

The worst case complexity of the algorithms is $\mathcal{O}(4^{(\Delta-3)k})$ multiplied by the complexity of the supporting algorithms, $\mathcal{O}(|\mathcal{V}| \cdot |\mathcal{A}| \cdot \log_{(1+|\mathcal{A}|/|\mathcal{V}|)} |\mathcal{V}|)$ [5], where k is the number of end nodes of resilient arcs (number of nodes of the graph in line 3 of FDP-NF, minus two – s and t), Δ is the maximum node degree of the graph G of the initial problem. The factor $(\Delta - 3)$ results from the fact that the same common node, which is not the end node of a shared resilient arc, may appear up to $\Delta - 3$ times in each branch of the generated problems.

In ASA the existence of a common node v will always result in 4 new problems (network graphs to be solved), while in AEA this will be the worst case; AEA will generate 4 problems only when both arcs in $\phi^+(v)$ are resilient. Also in ASA possibly new problems will be generated for every unreliable arc in $\delta^+(v)$, which will never happen in AEA. In AEA all unreliable arcs in $\delta^+(v)$ are removed together in Case 1. So although the worst case complexity of the algorithms is the same, AEA will tend to be more efficient than ASA (especially in networks with articulation nodes).

Example AEA. Consider the same example of Fig. 2. As in ASA the first execution of the main loop of line 4 leads to line 14 that begins the generation process of new elements to be added to $\mathcal{Q}$. None of the arcs forming the paths and emergent from c is resilient. Therefore, we generate three new problems. In the first of the new problems it is assumed that a shared resilient arc will leave c. Therefore, it will remove (c, d) and (c, t) from $\mathcal{A}$. Moreover, it will make (a, c) unreliable. The following two new problems will just remove one arc belonging to the paths and incident in c, *i.e.*, arc (a, c) or (b, c). Obviously the first two of those problems have no solution. The third problem has a solution, which is feasible

Algorithm 5 Algorithm Efficient Approach (AEA)

Require: Digraph $G = (\mathcal{V}, \mathcal{A})$; source node s; target node t; arc cost c_a such that $a \in \mathcal{A}$; set of unreliable arcs $\bar{\mathcal{S}} = \mathcal{A} \backslash \mathcal{S}$.

Ensure: (p, q) is the min-sum cost node-disjoint path pair from s to t except possibly at end nodes of *shared* resilient arcs, or $(\emptyset, \emptyset)$ if no solution was found.

Let $\mathcal{Q}$ be a queue of pairs of sets of arcs
Push $(\emptyset, \emptyset)$ into $\mathcal{Q}$ // Initial network is G
$(p, q) \leftarrow (\emptyset, \emptyset)$ // Initially: no solution and its cost is ∞
while $\mathcal{Q}$ is not empty **do**
    $(\hat{\mathcal{A}}, \hat{\mathcal{S}}) \leftarrow$ head of $\mathcal{Q}$
    Remove head of $\mathcal{Q}$
    $\mathcal{A}' \leftarrow \mathcal{A} \backslash \hat{\mathcal{A}}$ // Set of arcs of current problem (c.p.)
    $\bar{\mathcal{S}}' \leftarrow \mathcal{A}' \cap (\bar{\mathcal{S}} \cup \hat{\mathcal{S}})$ // Set of unreliable arcs of c.p.
    $(p', q') \leftarrow$ **FDP-NF**$(s, t, G'(\mathcal{V}, \mathcal{A}'), c, \bar{\mathcal{S}}')$ // or alternatively **AdvFDP-NF**
    if $(p', q') \neq (\emptyset, \emptyset)$ **then**
        if $c(p', q') < c(p, q)$ **then**
            if (p', q') is an admissible solution **then**
                $(p, q) \leftarrow (p', q')$ // updates (p, q)
            else // New problems
                $v \leftarrow firstCNode(p', q')$ // Case 1)
                $\hat{\mathcal{A}}_1 \leftarrow \hat{\mathcal{A}} \cup \chi(v)$ // Remove $\chi(v)$ arcs
                $\hat{\mathcal{S}}_1 \leftarrow \hat{\mathcal{S}} \cup \phi^-(v)$ // $\phi^-(v)$ now unreliable
                if $|\phi^+(v) \cap \bar{\mathcal{S}}'| \neq 0$ **then**
                    Push $(\hat{\mathcal{A}}_1, \hat{\mathcal{S}}_1)$ into $\mathcal{Q}$
                else // Both arcs in $\phi^+(v)$ are resilient
                    for all $a \in \phi^+(v)$ **do**
                        Push $(\hat{\mathcal{A}}_1 \cup \{a\}, \hat{\mathcal{S}}_1)$ into $\mathcal{Q}$
                    end for
                end if
                for all $a \in \phi^-(v)$ **do** // Case 2)
                    Push $(\hat{\mathcal{A}} \cup \{a\}, \hat{\mathcal{S}})$ into $\mathcal{Q}$
                end for
            end if
        end if
    end if
end while
return path pair (p, q) // No solution if $(p, q) = (\emptyset, \emptyset)$

and identical to the solution obtained using ASA. Therefore, the algorithm will end after executing the main loop of line 4 for the fourth time. Notice that in ASA the main loop had to be executed five times.

However, for the same example network of Fig. 2, if AdvFDP-NF is used instead of FDP-NF, the optimal solution would be obtained by both AEA and ASA in the first iteration – see Fig. 4.

6.3. Improving the efficiency of both algorithms

In ASA or AEA, new problems will always have a cost equal or greater than the cost of the originating problem. The correctness of this statement is clear if one recalls that every time a new problem is generated, at least one arc is pruned and possibly a resilient arc is made unreliable. If the cost of the current best solution (p, q) (assuming it is not ∞) is smaller than the cost of the path pair (p', q'), no problems generated using (p', q') will improve the present solution. This knowledge is used (see line 11 of ASA and 11 of AEA) to prevent generating problems that will not improve the current best solution of ASA or AEA.

Furthermore, the cost of the originating problem is also kept as the second (third) part of each element stored in the queue used in ASA (AEA). This information can be utilized to avoid using FDP-NF (in both algorithms) for solving problems whose cost will be equal or greater than the cost of the current best solution. So a new line: "**if** $(p, q) = (\emptyset, \emptyset) \vee$ cost of originating problem of the current problem $< c(p, q)$ **then**" should be after line 6 of ASA and 6 of AEA. Hence, the current problem (the one just removed from the head of the queue) will only be solved if this condition is true.

Additionally instead of making a BFS search in the generated problems, ASA and AEA can store the new problems in a priority queue, ordered by the increasing cost of the originating problems. This will result in the need to solve less problems, because it is likely that with this approach an admissible solution of lower cost will be found earlier than using the BFS search.

Finally, and in order to help identifying problems with no solution, the first step will be the determination of a maximally node disjoint path pair of min-sum cost [37]. If in the resulting path pair:

- an unreliable arc is shared, no solution can be found and the algorithm ends;
- no resilient arc is emergent or incident in any of the intermediate common nodes, there is no solution and the algorithm ends;

Also, if the obtained maximally node disjoint path pair is an admissible solution to SNDP^{2}SRA its cost can also be used as an upper-bound to the cost of the optimal solution, and thus avoiding generating new problems that will have a higher cost than this upper-bound. Obviously, this upper-bound improvement is weaker than the improvement that uses a priority queue.

Table 1: Networks from [38], where n is the number of nodes and m the number of edges

Network	n	m	m/n	biconnected
atlanta	15	22	1.47	yes
newyork	16	49	3.06	yes
nobel-germany	17	26	1.53	yes
geant	22	36	1.64	yes
nobel-eu	28	41	1.46	yes
india35	35	80	2.29	yes
pioro40	40	89	2.23	yes
germany50	50	88	1.76	yes
france	25	45	1.80	no
ta2	65	108	1.66	no

7. Numerical Results

For the performance evaluation of the proposed algorithms ten networks from [38] were considered. Their brief description can be found in Table 1; the integer part of the first module cost associated with each edge was taken as the edge cost. As noted in Table 1, eight of the ten considered networks are bi-connected, but france and ta2 have each two articulation nodes.

Network operators do not disclose data allowing to state which is the most realistic percentage of resilient edges. In the scenario where resilient edges represent third party networks that interconnect sub-networks of a single operator, the percentage of resilient edges may be small. On the other hand, in general the number of resilient edges may be very high, because protection is commonly used in optical networks. Nevertheless certain edges may be prone to dual faults—which may result in the failure of the corresponding direct connection albeit the underlying 1+1 protection—and as such may be considered unreliable edges. In this case to increase end-to-end resiliency, node-disjoint paths sharing resilient edges may used. Hence, the number of reliable edges was considered equal to 5%, 10%, 15%, 25%, 50% and 80%, to cover different application scenarios. For each of these values 20 different seeds were considered generating 20 networks which differ in the resilient edges distribution. The corresponding directed graphs were then created.

The proposed algorithms will calculate a path pair per request, and the network state may change between requests. However, to evaluate the algorithms performance the problem was solved for all node pairs (s, t) $(s \neq t)$ considering a static network state, defined by the topology (with resilient and unreliable edges) and the edges' cost.

The CPU time (in milliseconds) per node pair, with 95% confidence intervals, were calculated. Figs. 6 and 8 illustrate the obtained results. The CPU time required for solving these problems using CPLEX 12.5 [39] is also given in the figures. All results were obtained in a Dell Optiplex 7010, Intel Core i7-3770 CPU (four core, 3.40GHz, 8MB), with 16GB of RAM, referred to as "Desktop" in the figures' captions.

In Fig. 6 the results considering 15% resilient edges are presented for all the networks. It

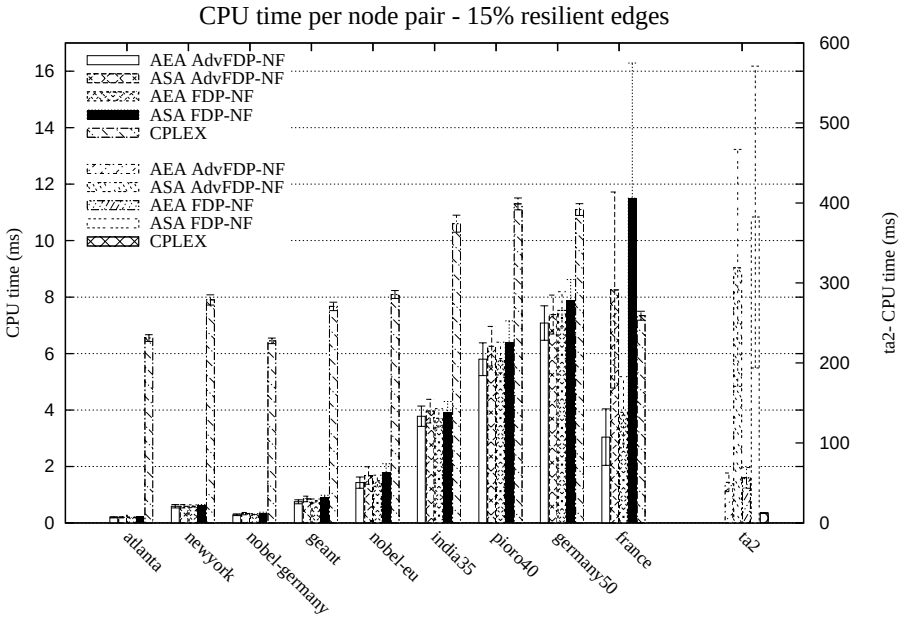

Figure 6: CPU time per node pair (milliseconds) in Desktop, considering 15% resilient edges.

can be observed that the CPU time used by the algorithms grows with the number of nodes and edges; the same happens with CPLEX, but in a much less pronounced way. The relative behavior of the algorithms for 5% and 10% resilient edges, is similar to the one observed for 15% resilient edges, so we omit the respective figures. We only would like to point out that the CPU time decreases with the number of resilient edges: for 5% resilient edges the average CPU time is below 4 ms for ta2 and below 2 ms for all other networks. For 10% resilient edges the average CPU time is below 20 ms for ta2 and below 4 ms for all other networks. Solving with CPLEX for ta2 network required 10 ms and 11 ms for 5% and 10% resilient edges, respectively; for the other networks it took between 6 ms and 9 ms for 5% resilient edges, and between 6 ms and 10 ms for 10% resilient edges.

Regarding the relative performance of AEA and ASA (for the same supporting algorithm FDP-NF or AdvFDP-NF) the algorithms present similar CPU times in the case of the biconnected networks, with a consistent, albeit slight, advantage for AEA. However, in the case of the networks with articulation nodes (france and ta2) AEA is clearly the most efficient approach.

The CPU time of ASA and AEA algorithms is directly linked to the size of the networks, the percentage of resilient edges, and the number of solved subproblems (the number of times AdvFDP-NF or FDP-NF are executed). It was observed that for the biconnected networks the average number of subproblems is close to one, which means that in most cases the supporting algorithm (FDP or FDP-NF) solves the SNDP^{2}SRA to optimality—see the results for pioro40 network in Fig. 7. Only for the non-biconnected networks is the average number of solved subproblems larger than two, as shown for france network in Fig. 7 (y axis on the right).

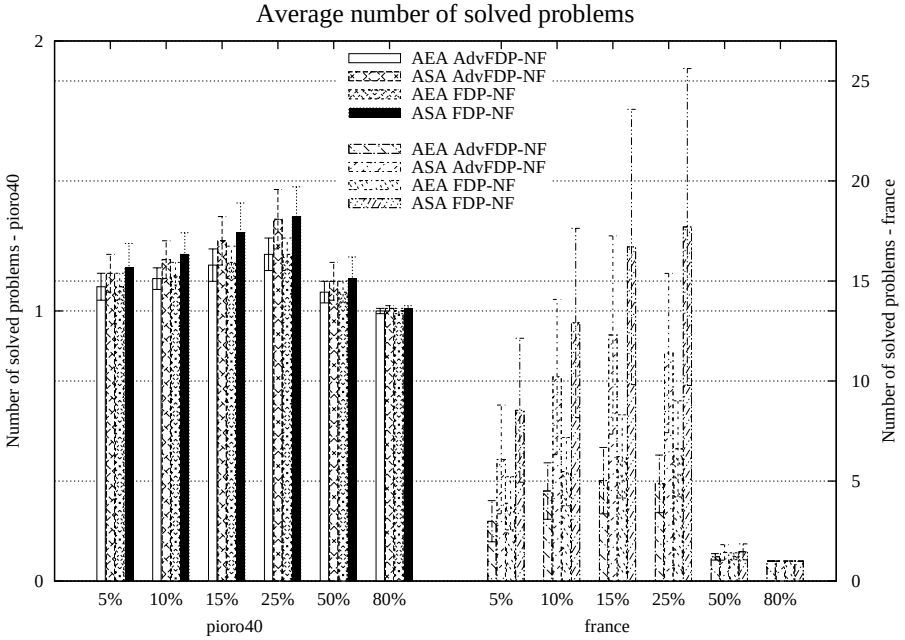

Figure 7: Average number of solved subproblems for pioro40 and france networks.

Let us now comment on the significant advantage AEA has over ASA for france and ta2 networks. When path pairs are calculated between specific end nodes in different biconnected components, those paths will have to share a node or two. An articulation node (or both articulation nodes, depending on the location of the end nodes) will appear in every candidate solution for those node pairs. If a single resilient arc is adjacent to the common articulation node, and no solution exists, ASA will take much longer than AEA to verify this fact. Moreover, when this articulation node is the first common node, AEA generates problems where all unreliable arcs emergent from the common articulation node are removed, while ASA removes the emergent edges, one by one. In this case AEA will consistently generate (and solve) less problems than ASA (see sub-section 6.2).

The fact that AEA needs to solve on average much less problems than ASA for networks with articulation nodes is clearly seen in Fig. 7 for france network: the number of solved problems, for 5%-25% resilient edges is much higher in ASA than in AEA. When the number of resilient edges is 50%, there is a high probability that at least one resilient edge (two symmetric resilient arcs) will be adjacent to every node. And for 80% resilient edges this will be even more probable, hence the number of solved subproblems needed to obtain an optimal solution is close to one, because there is a great probability that the first solution by either FDP-NF or AdvFDP-NF is an admissible solution, and hence the optimal one.

In france network only three edges belong to the two smaller biconnected components. In the case of ta2 network (which has more than twice the number of nodes and edges of france network) the two smaller biconnected components have one and six edges, respectively. This is the reason why CPU times for ta2 network, for 5%-25% resilient edges, are much larger than for the other tested networks (note the second y axis in Fig. 6).

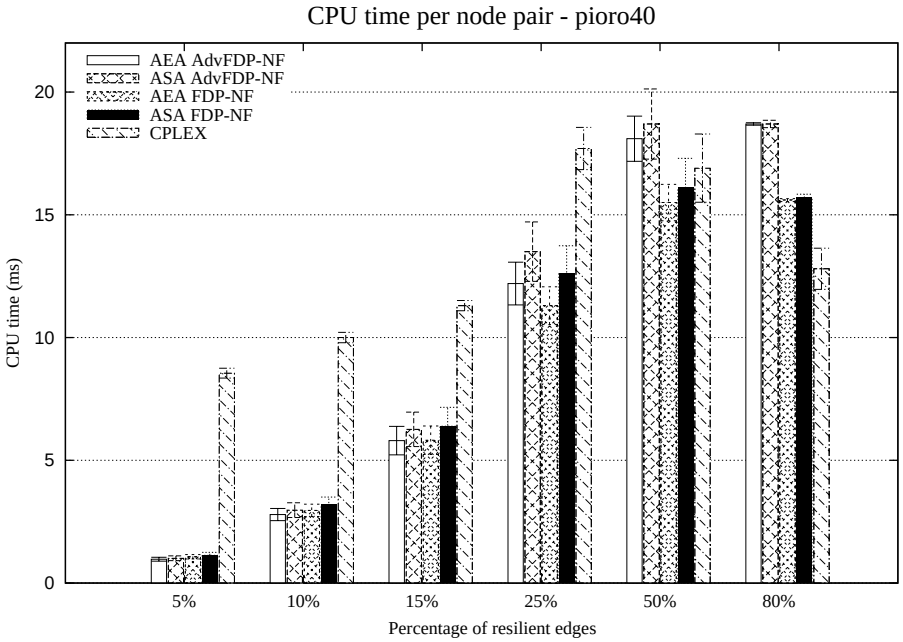

Figure 8: CPU time per node pair (milliseconds) in the Desktop, for pioro40 network.

Regarding the impact of selecting FDP-NF or AdvFDP-NF as supporting algorithm, it was verified that for a smaller percentage of resilient edges AdvFDP-NF tends to be more efficient as shown in Fig. 7. However, when the number of resilient edges increases, FDP-NF presents lower CPU time than AdvFDP-NF, as can be seen in Fig. 8 for 50% and 80% resilient edges (for pioro40 network). Recall that FDP-NF divides all nodes considered unreliable, and then proceeds to calculate arc-disjoint path pairs between each vertex in sources to each vertex in sinks (see sub-section 5.1 for a definition of sources and sinks). On the other hand, AdvFDP-NF does not require to divide any node in the beginning, as it calculates node-disjoint path pairs between each vertex in sources to each vertex in sinks. However, in AdvFDP-NF nodes have to be divided while calculating the mentioned node-disjoint paths. If the percentage of resilient edges is small AdvFDP-NF needs to calculate a relatively small number of node-disjoint path pairs, hence it makes few node divisions. In contrast, FDP-NF has to divide a large number of nodes in the beginning. Therefore AdvFDP-NF requires less CPU time than FDP-NF. However if the percentage of resilient edges is high, AdvFDP-NF needs to calculate a larger number of node-disjoint path pairs, therefore AdvFDP-NF has to make more node divisions, and becomes slightly less efficient, in terms of CPU time, than FDP-NF. Nevertheless using AdvFDP-NF instead of FDP-NF (for the same algorithm AEA or ASA) results in a lower average number of solved problems (although confidence intervals overlap)—see Fig. 7.

Concerning the algorithms performance when compared to the efficiency of using the MIP approach, for the biconnected networks with less than 30 nodes the algorithms present much smaller CPU times than CPLEX. As the number of resilient edges increases the performance of the algorithms may reduce, when compared to CPLEX, due to the increasing number of

node-disjoint (AdvFDP-NF) or edge-disjoint (FDP-NF) path pairs required to create the modified graph, in the cases where the average number of solved problems does not change significantly because it was already very low (less than two). Still, up to 25% resilient edges AEA presents lower CPU times than CPLEX for all networks with the exception of ta2. An example of this behavior can be seen in Fig. 8, where CPU times for all considered percentages of resilient edges are shown for the case of pioro40 network. The CPU time required by the algorithms for solving SNDP^{2}SRA in ta2 network is larger than required by CPLEX for 10%-80% resilient edges. Yet, even for that network, the CPU time of AEA is still compatible with a network management system (where CPLEX may not be available), because the average CPU time per node pair was less than 110 milliseconds in the used Desktop. What is more, it is important to notice that the presented optimization problems are to be solved during network operations and not during a planning and designing phase. What is more, they occur independently in different nodes of a network. It implies that in practice it would be difficult to employ commercial MIP solvers to handle those problems. Therefore, easily implementable specialized algorithms for such problems are of high interest.

The average CPU time per node pair was also calculated in a Path Computation Element (PCE), model UNICOM-V5, G2_LE CPU (PowerPC compatible core) with 330 MHz core clock, 128 MB of RAM. A compiled dynamic library in pq2 was created, and linked to a test program calling AEA or ASA. The CPU time obtained in the considered PCE can be seen in Fig. 9 for 15% resilient edges. From these results we may conclude that, in the PCE, the algorithm with the best performance is still AEA, namely due to the CPU times observed in the case of france network. From the experiments in Desktop it was clear that for ta2 network ASA had a very poor performance, when compared to AEA. Therefore in the case of the PCE only results for AEA and the two supporting algorithms are given in Fig. 9. It can be concluded that for the biconnected smaller networks (less than 30 nodes) AEA (using any of the supporting algorithms) can be used in the GMPLS control plane. For larger networks the use of the proposed algorithms will probably only be possible assuming that the PCE is in the management plane, because the CPU time per node pair can be as large as 5 seconds when the considered PCE is used to compute a pair of paths in ta2 network for 15% resilient edges.

Summarizing, the algorithm with the overall best performance is AEA. Regarding the supporting algorithms, AdvFDP-NF should be used when the percentage of resilient edges is small ($< 50\%$) and FDP-NF should be used if that percentage is large ($\geq 50\%$). When the number of resilient edges is high and the networks have more than 30 nodes, we can either employ CPLEX (if we can afford it) or use AEA with FDP-NF.

8. Conclusion

In a multilayer scenario, each layer may have its own protection mechanisms. Considering resilient routing in a MPLS network over a partially protected optical layer, efficient algorithms are required to ensure that protection duplications are avoided.

In the paper we were facing a problem of finding the shortest node-disjoint pair of paths that can share resilient arcs and cannot contain loops. We modeled the problem by a mixed

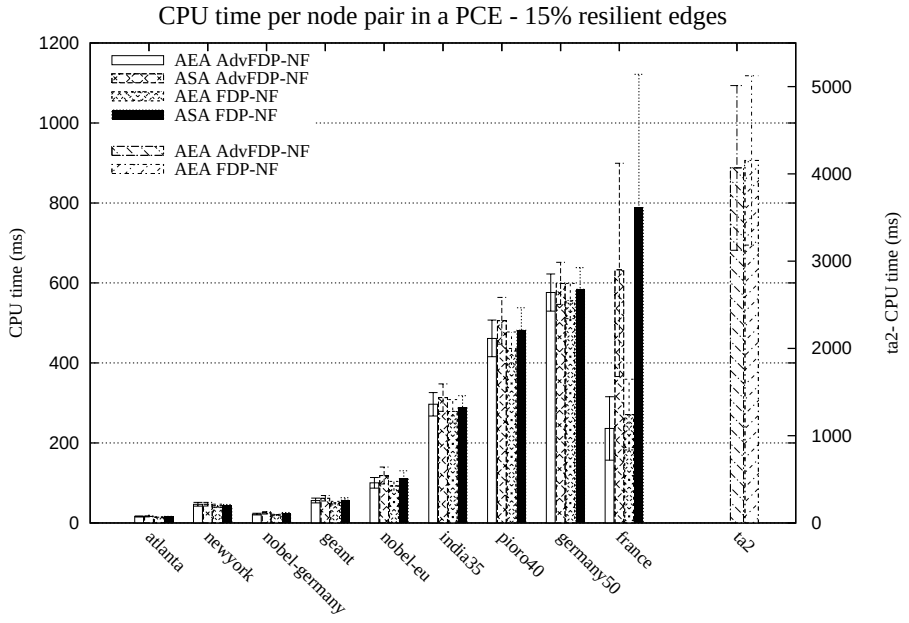

Figure 9: CPU time per node pair (milliseconds) in a PCE using a dynamically linked shared object library, considering 15% resilient edges.

integer program (MIP), and pointed a number of key features that characterize its feasible solutions. Moreover, we directly listed a number of special features that differentiate our problem from other similar problems considered in the literature.

In total we presented four different algorithms solving the considered problem. The algorithms have identical worst case complexity, but differ from each other with respect to their running times, especially in networks with articulation nodes. The algorithms have been implemented and tested on real-world size telecommunication networks. The algorithms proved to be efficient and return solutions in time comparable or better than time needed by commercial MIP solvers. Moreover, they proved to be applicable as practical approaches for finding primary-backup path pairs in industrial solutions.

Acknowledgement

The work of Teresa Gomes has been partially supported by PT Inovação R&D Project "End to end Protection considering SRLGs – II", by the Portuguese Foundation for Science and Technology under project grant PEst-OE/ EEI/UI308/2014 and by ICIS Project CENTRO-07-0224-FEDER-002003. The work of Mateusz Żotkiewicz has been supported by National Science Center (Poland) under grant 2011/01/B/ST7/02967.

References

[1] T. Gomes, M. Żotkiewicz, Finding the shortest node-disjoint pair of paths that are allowed to share resilient arcs, in: Proc. of 5th International Workshop on Reliable Network Design and Modeling, 2013,

pp. 1055–1064. doi:10.1109/ICUMT.2013.6798424.
[2] D. Tipper, Resilient network design: challenges and future directions, Telecommunication Systems (2013). Published on line: 15 August 2013.
[3] M. Pióro, D. Medhi, Routing, Flow, and Capacity Design in Communication and Computer Networks, Morgan Kaufman Series on Networking, Morgan Kaufman, San Francisco, California, USA, 2004.
[4] A. Urra, E. Calle, J. L. Marzo, Enhanced protection using shared segment back-ups in a multiservice GMPLS-based networks, in: IEEE Symposium on Computers and Communications, IEEE Computer Society, Los Alamitos, CA, USA, 2005, pp. 752–757. doi:http://doi.ieeecomputersociety.org/10.1109/ISCC.2005.65.
[5] M. Żotkiewicz, W. Ben-Ameur, M. Pióro, Finding failure-disjoint paths for path diversity protection in communication networks, IEEE Communications Letters 14 (2010) 776 –778.
[6] J.-P. Vasseur, M. Pickavet, P. Demeester, Network Recovery – Protection and Restoration of Optical, SONET-SDH, IP, and MPLS, Elsevier, 2004.
[7] C. Awad, B. Sansò, Network reliability under mixed IP and optical protection, in: Design of Reliable Communication Networks, 2005. (DRCN 2005). Proceedings.5th International Workshop on, 2005, pp. 8 pp.–.
[8] S. Kartalopoulos, Free Space Optical Networks for Ultra-Broad Band Services, Springer-Verlag, 2011.
[9] Y. Li, M. Pioro, V. Angelakisi, Design of cellular backhaul topology using the fso technology, in: Optical Wireless Communications (IWOW), 2013 2nd International Workshop on, 2013, pp. 6–10.
[10] A. Farrel, J.-P. Vasseur, J. Ash, A path computation element (PCE)-based architecture, ITEF RFC 4655, 2006.
[11] Y. Iizawa, S. Araki, S. Ishida, I. Nishioka, K. Shimada, H. Hasegawa, K. ichi Sato, PCE-based fast path control in multi-domain photonic networks, Optical Switching and Networking 10 (2013) 32 – 43.
[12] E. K. Cetinkaya, J. P. Sterbenz, A taxonomy of network challenges, in: 9th International Conference on Design of Reliable Communication Networks (DRCN), Budapest, Hungary, 2013, pp. 322–330.
[13] B. Jaumard, M. Bui, B. Mukherjee, C. S. Vadrevu, {IP} restoration vs. optical protection: Which one has the least bandwidth requirements?, Optical Switching and Networking 10 (2013) 261 – 273.
[14] P.-H. Ho, J. Tapolcai, T. Cinkler, Segment shared protection in mesh communications networks with bandwidth guaranteed tunnels, IEEE/ACM Transactions on Networking 12 (2004) 1105–1118.
[15] J. W. Suurballe, R. E. Tarjan, A quick method for finding shortest pairs of disjoint paths, Networks 14 (1984) 325–336.
[16] D. Xu, Y. Chen, Y. Xiong, C.Qiao, X. He, On the complexity of and algorithms for finding the shortest path with a disjoint counterpart, IEEE/ACM Transactions on Networking 14 (2006) 147–158.
[17] L. Guo, H. Shen, On the complexity of the edge-disjoint min-min problem in undirected graphs, in: 3rd International Conference on Computer Research and Development (ICCRD), volume 2, Shanghai, China, 2011, pp. 150–154. URL: http://dx.doi.org/10.1109/ICCRD.2011.5764102.
[18] M.-L. Gan, S.-Y. Liew, Effective algorithms for finding optimum pairs of link-disjoint paths in $\alpha + 1$ path protection, Telecommunication Systems 52 (2013) 783–797.
[19] P. Laborczi, J. Tapolcai, P.-H. Ho, T. Cinkler, A. Recski, H. T. Mouftah, Algorithms for asymmetrically weighted pair of disjoint paths in survivable networks, in: T. Cinkler (Ed.), Proceedings of Design of Reliable Communication Networks (DRCN), Budapest, Hungary, 2001, pp. 220–227.
[20] T. Gomes, J. Craveirinha, L. Jorge, An effective algorithm for obtaining the minimal cost pair of disjoint paths with dual arc costs, Computers & Operations Research 36 (2009) 1670–1682.
[21] J. Y. Yen, Finding the k shortest loopless paths in a network, Management Science 17 (1971) 712–716.
[22] E. Martins, M. Pascoal, J. Santos, Deviation algorithms for ranking shortest paths, International Journal of Foundations of Computer Science 10 (1999) 247–263.
[23] D. Xu, Y. Xiong, C. Qiao, Novel algorithms for shared segment protection, IEEE Journal on Selected areas in Communications 21 (2003) 1320–1331.
[24] P. Pan, G. Swallow, A. Atlas, Fast reroute extensions to RSVP-TE for LSP tunnels, IETF RFC 4090, 2005.
[25] M. Kodialam, T. V. Lakshman, Dynamic routing of locally restorable bandwidth-guaranteed tunnels

using aggregated link usage information, in: IEEE INFOCOM 2001, 2001, pp. 376–385.
[26] M. Kodialam, T. V. Lakshman, Restorable dynamic quality of service routing, IEEE Communications Magazine 40 (2002) 72–81.
[27] C. Qiao, D. Xu, Distributed partial information management DPIM schemes for survivable networks – part I, in: IEEE INFOCOM 2002, 2002, pp. 302–311.
[28] T. Cinkler, R. Kosznai, P. Soproni, K. Németh, GSP, the generalised shared protection, in: 9th International Conference on Design of Reliable Communication Networks – DRCN 2013, Budapest, Hungary, 2013, pp. 195–202.
[29] W. Grover, D. Stamatelakis, Cycle-oriented distributed pre-configuration: ring-like speed with mesh-like capacity for self-planning network restoration, in: Proc. IEEE International Conf. Commun. (ICC'98), Atlanta, USA, 1998, pp. 537–543.
[30] R. Asthana, Y. Singh, W. Grover, p-cycles: An overview, IEEE Communication Surveys and tutorials 12 (2010) 97–111.
[31] W.-D. Zhong, F. Zhang, An overview of p-cycle based optical multicast protection approaches in mesh WDM networks, Optical Switching and Networking 8 (2011) 259 – 274.
[32] X. Zhang, K. Li, L. Bian, Towards the maximum resource sharing degree for survivable IP/MPLS over WDM mesh networks, Optical Switching and Networking 11, Part B (2014) 177 – 188.
[33] E. Mannie, Generalized multi-protocol label switching (GMPLS) architecture, 2004. URL: `http://tools.ietf.org/pdf/rfc3945.pdf`.
[34] J. Lang, Y. Rekhter, D. Papadimitriou, (Eds), RSVP-TE extensions in support of end-to-end generalized multi-protocol label switching (GMPLS) recovery, ETF RFC 4872, 2007.
[35] L. Berger, Generalized multi-protocol label switching (GMPLS) signaling resource reservation protocol-traffic engineering (RSVP-TE) extensions, IETF RFC 3473, 2003.
[36] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, RSVP-TE: Extensions to RSVP for LSP tunnels, IETF RFC 3209, 2001.
[37] R. Bhandari, Survivable Networks, Algorithms for Diverse Routing, Kluwer Academic Publishers, Norwell, Massachusetts, USA, 1999.
[38] S. Orlowski, M. Pióro, A. Tomaszewski, R. Wessäly, SNDlib 1.0–Survivable Network Design library, Networks 55 (2010) 276–286.
[39] IBM ILOG CPLEX Optimization Studio V12.5, IBM, 2012.