Availability-based path selection and network vulnerability assessment

Yang, S; Trajanovski, S; Kuipers, FA

DOI
10.1002/net.21652

Publication date
2015

Document Version
Accepted author manuscript

Published in
Networks: an international journal

Citation (APA)
Yang, S., Trajanovski, S., & Kuipers, FA. (2015). Availability-based path selection and network vulnerability assessment. *Networks: an international journal*, *66*(4), 306-319. https://doi.org/10.1002/net.21652

Availability-Based Path Selection and Network Vulnerability Assessment

Song Yang, Stojan Trajanovski and Fernando A. Kuipers
Delft University of Technology, The Netherlands
{S.Yang, S.Trajanovski, F.A.Kuipers}@tudelft.nl

***Abstract*—In data-communication networks, network reliability is of great concern to network operators and customers, since (1) the connection requested by the customer should obey the agreed-upon availability, otherwise the network operator or service provider may face liability costs as stipulated in the Service Level Agreement (SLA), and (2) it is important to determine the most vulnerable parts of a network to gain insight into where and how the network operator can increase the network reliability.**

In this paper, we first study the problem of establishing a connection over at most k (partially) link-disjoint paths and for which the availability is no less than δ ($0 < \delta \leq 1$). We analyze the complexity of this problem in generic networks, Shared-Risk Link Group (SRLG) networks and multi-layer networks. We subsequently propose a polynomial-time heuristic algorithm and an exact Integer Non-Linear Program (INLP) for availability-based path selection. The proposed algorithms and two existing heuristic algorithms are compared in terms of acceptance ratio and running time. Subsequently, in three aforementioned types of networks, we study the problem of finding a (set of) network cut(s) for which the failure probability of its links is biggest.

***Index Terms*—Availability, Routing, Survivability, SRLG networks, Multi-layer networks, Min-cut.**

I. INTRODUCTION

Due to the importance of data-communication networks, even short service disruptions may result in significant economic loss. Hence, survivability mechanisms to protect connections are called for. For instance, by allocating a pair of link-disjoint paths (instead of only one unprotected path), data are transported by the primary path, and upon link failure, can be switched to the backup path.

Ideally, a survivability mechanism should also take into account the reliability of links. For instance, if both primary and backup paths contain links that have a high probability to become unavailable, then proper protection cannot be provided. Connection availability, a value between 0 and 1, is therefore important and refers to the probability that a connection (including its survivability mechanism) is in the operating state during the requested life-time of the connection.

However, a survivability mechanism that does not allow for more than 2 link-disjoint paths for each connection may still fail to satisfy the customer's availability requirement and $k > 2$ link-disjoint paths may be needed. Obviously, the bigger k is, the greater the availability of the connection could be, but also the greater the resource consumption (e.g., bandwidth) and hence price. This paper first deals with the Availability-Based Path Selection (ABPS) problem, which is to establish a connection over at most $k > 0$ (fully or partially) link-disjoint paths, for which the availability is at least δ ($0 < \delta \leq 1$).

Apart from considering how to provide a reliable connection to customers, it is also important for network operators to determine the most vulnerable part of the network, i.e., a subset of links with highest failure probability whose removal will disconnect the network. The network operator could then replace/strengthen those links in order to increase the network reliability. Hence, this paper also tackles this so-called Network Vulnerability Assessment (NVA) problem, which is to find a set of network cuts for which the failure probability of the links in a cut belonging to that set is highest.

Our key contributions are as follows:

- We consider the Availability-Based Path Selection (ABPS) problem in generic networks, Shared-Risk Link Group (SRLG) networks and multi-layer networks.
- We prove that, in general, the ABPS problem cannot be approximated in polynomial time.
- We propose a polynomial-time heuristic algorithm and an exact Integer Non-Linear Program (INLP) to solve the ABPS problem.
- We compare, via simulations, the proposed algorithms with two existing algorithms in terms of performance and running time.
- We consider the Network Vulnerability Assessment (NVA) problem in generic networks, SRLG networks and multi-layer networks.

The remainder of this paper is organized as follows. Related work is presented in Section II. Section III explains the calculation of availability for different path types: unprotected path, k fully link-disjoint and k partially link-disjoint. In Section IV, we formally define the Availability-Based Path Selection (ABPS) problem in generic networks and analyze its complexity. In Section V and VI, we consider the ABPS problem in SRLG networks and multi-layer networks, respectively. Section VII presents our heuristic routing algorithm and an exact INLP. Section VIII provides our simulation results. In Section IX, we study the Network Vulnerability Assessment problem in the three aforementioned types of networks. We conclude in Section X.

II. RELATED WORK

Availability-aware routing under both static and dynamic traffic demands has been extensively investigated [1], [2], [3], [4], [5], [6]. When the traffic matrix is given in advance (static traffic), Zhang *et al.* [3] present a mathematical model to compute availability for different protection types (unprotected, dedicated protection and shared protection) for a given static traffic matrix. Furthermore, an Integer Linear Program (ILP) and a heuristic algorithm are proposed to find availability-aware paths. Tornatore *et al.* [4] address the availability design problem: to accommodate a given traffic matrix by using shared/dedicated protection paths. Song *et al.* [5] propose an availability-guaranteed routing algorithm, where different protection types are allowed. They define a new cost function for computing a backup path when the unprotected path fails to satisfy the availability requirement. She *et al.* [1] prove that for dedicated protection, finding two link-disjoint paths with maximal reliability (availability) is NP-hard. They also propose two heuristics for that problem. Luo *et al.* [6] analyze the problem of protection with different reliability, which is to find one unprotected path or dedicated protection path such that the cost of the whole path is minimized and the reliability requirement is satisfied. They subsequently propose an exact ILP as well as two approximation algorithms. However, the reliability (availability) calculation in [6] is different from the aforementioned papers, and assumes a single-link failure model. Assuming each link in the network has a failure probability (=1-availability), Lee and Modiano [2] minimize the total failure probability of unprotected, partially link-disjoint and fully link-disjoint paths by establishing INLPs. They further transform the proposed INLPs to ILPs by using linear approximations.

Different from the aforementioned articles, we target a more general problem, which is to find at most k (fully or partially) link-disjoint paths for which the availability requirement is satisfied.

From the perspective of network reliability calculation, assuming each link in the network is associated with a failure probability value (=1-availability), Provan and Ball [7] prove the problem of computing the probability that the network stays connected is #P-complete[1]. Karger [9] proposes a Fully Polynomial Randomized Approximation Scheme (FPRAS) to solve this problem. There is also work focusing on how to mathematically model the availability of various network topologies or different protection segments/paths. Zou *et al.* [10] investigate how to mathematically calculate the availability of different types of network topologies, e.g., tree topology, double star, crown or triple star. Tornatore *et al.* [11] mathematically model the availability of segment protection (SP). In segment protection, a working path (WP) can be partitioned into several working segments (WSs) and each WS is protected by a backup segment (BS). Moreover, they consider two SP cases, namely (1) overlap SP, where different WSs can share the same link, and (2) no-overlap SP, where WSs are fully link-disjoint. By expressing the dual-link failure via a continuous time Markov chain, Mello *et al.* [12] approximately estimate the (un)availability of the shared protection path.

[1]Valiant [8] shows that problems in this class are at least as hard as NP-complete problems.

Regarding SRLG networks, Hu [13] proves that the problem of finding 2 SRLG-disjoint paths is NP-hard. To solve it, Hu [13] presents an exact ILP and Xu *et al.* [14] propose a trap-avoidance heuristic algorithm. However, the SRLG-disjoint routing problem is not the same as the one studied in this paper, due to Eq. (11) in Section V. Hence, the algorithms in [13] [14] cannot be used to effectively solve our problem.

In generic networks, the (s, t) Min-Cut problem refers to partitioning the network into two disjoint subsets such that nodes s and t are in different subsets and the total weight of the cut links is minimized. According to [15], this problem can be solved by finding the maximum flow from s to t. There is also a lot of work on the Min-Cut problem with no specified node pairs (s, t). An enumeration of classical algorithms to solve the Min-Cut problem can be found in [16].

III. Connection Availability

The availability of a system is the fraction of time the system is operational during the entire service time. Like [1], [2], [3], we first assume that, in generic networks, the links' availabilities are uncorrelated/independent. If a connection is carried by a single (unprotected) path, its availability is equal to the path availability; if it is protected by $k \geq 2$ disjoint paths, the availability will be determined by these k protection paths. The availability A_j of a network component j can be calculated as [10]:

$$A_j = \frac{MTTF}{MTTF + MTTR} \tag{1}$$

where $MTTF$ represents Mean Time To Failure and $MTTR$ denotes Mean Time To Repair. We assume that the link availability is equal to the product of availabilities of all its components (e.g., amplifiers).

A. Link Failure Scenarios

For simplicity, suppose there are two (fully) link-disjoint paths p_1 and p_2, and the availability of link l is denoted as A_l $(= 1 - failure\ probability)$, where $0 < A_l \leq 1$, then their total availability A_{FD}^2 can be computed based on the following scenarios:

- **Single-link failure**: Here it is assumed that all the links in the network have very low failure probability. In this context, a path p's availability (denoted by A_p) is equal to its lowest traversed link availability (highest failure probability), i.e., $A_p = \min_{l \in p} A_l$. Using two disjoint paths (which is a conventional survivability mechanism) will therefore lead to a total connection availability of 1. However, this approach only works when all the links are highly reliable. In Appendix A, we will address the ABPS problem under the single-link failure scenario.
- **Multiple link failures:** This is a more general scenario where at one certain point in time, several links in the network may fail simultaneously. Hence, for a path p, its availability A_p should take into account all its links' availabilities, i.e., $A_p = \prod_{l \in p} A_l$. Consequently, $A_{FD}^2 =$

$1-(1-A_{p_1})(1-A_{p_2})$, which indicates the probability that at least one path is available. In this paper, we assume multiple link failures may occur.

B. End-to-End Path Availability

If a path p contains the links l_1, l_2, l_3,..., l_m, and their corresponding availabilities are denoted by A_{l_1}, A_{l_2}, A_{l_3},..., A_{l_m}, then the availability of this path (represented by A_p) is equal to[2]:

$$A_p = A_{l_1} \cdot A_{l_2} \cdot A_{l_3} \cdots A_{l_m} \tag{2}$$

If we take the $-log$ of the link availabilities, finding the path with the highest availability turns into a shortest path problem.

When, for a single connection, there are $k \geq 2$ link-disjoint paths p_1, p_2,..., p_k with availabilities represented by A_{p_1}, A_{p_2},..., A_{p_k}, the connection availability calculation can be divided into two cases, namely: (1) fully link-disjoint paths: these k paths have no links in common, and (2) partially link-disjoint paths: at least two of these k paths traverse at least one same link. In case (1), the availability (represented by A^k_{FD}) is:

$$A^k_{FD} = 1-\prod_{i=1}^{k}(1-A_{p_i}) = \sum_{i=1}^{k} A_{p_i} - \sum_{0<i<j\leq k} A_{p_i}\cdot A_{p_j} + \sum_{0<i<j<u\leq k} A_{p_i}\cdot A_{p_j}\cdot A_{p_u} + \cdots + (-1)^{k-1}\prod_{i=1}^{k} A_{p_i} \tag{3}$$

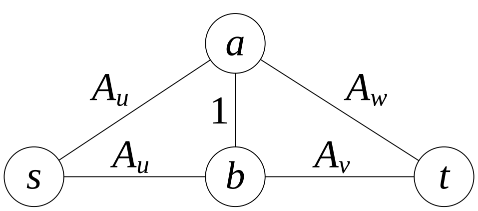


Fig. 1: Network with different link availability values.

If we use Eq. (3) to calculate the availability for the partially link disjoint case, the availability of the overlapping links will be counted more than once. To amend this, we use a new operator $\circ$, which is defined as follows:

$$X_1 \cdot X_2 \cdots X_k \circ Y = \begin{cases} \prod_{i=1}^{k} X_i & \text{if} \quad \exists X_i = Y \\ \prod_{i=1}^{k} X_i \cdot Y & \text{otherwise} \end{cases} \tag{4}$$

where $X_1, X_2, ..., X_k$ and Y represent different link availabilities. Therefore, the availability (represented by A^k_{PD}) of k partially link-disjoint paths can be calculated as follows:

[2]A network having node and link availabilities can be transformed to a directed network with only link availabilities, as done in [17]. Therefore, we assume the nodes have availability 1 in this paper.

$$\begin{aligned} A^k_{PD} &= 1-\coprod_{i=1}^{k}(1-A_{p_i}) \\ &= 1-(1-A_{p_1})\circ(1-A_{p_2})\circ\circ\circ(1-A_{p_k}) \\ &= \sum_{i=1}^{k} A_{p_i} - \sum_{0<i<j\leq k} A_{p_i}\circ A_{p_j} + \\ & \sum_{0<i<j<u\leq k} A_{p_i}\circ A_{p_j}\circ A_{p_u} + \cdots + (-1)^{k-1}\coprod_{i=1}^{k} A_{p_i} \end{aligned} \tag{5}$$

where $\coprod$ is used to denote the $\circ$ operations of different sets. Let us use an example to describe the difference between case (1) and case (2), where k is set to 2 for simplicity. In Fig. 1 where the link availability is labeled on each link, paths $s-a-t$ and $s-b-t$ are fully link disjoint. According to Eq. (3), their availability is equal to:

$$\begin{aligned} &1-(1-A_u\cdot A_w)\cdot(1-A_u\cdot A_v) \\ =&A_u\cdot A_w + A_u\cdot A_v - A_u^2\cdot A_w\cdot A_v \end{aligned} \tag{6}$$

On the other hand, paths $s-a-t$ and $s-a-b-t$ are only partially link disjoint. According to Eq. (5), the connection availability can be calculated as follows:

$$\begin{aligned} &1-(1-A_u\cdot A_w)\circ(1-A_u\cdot A_v) \\ =&A_u\cdot A_w + A_u\cdot A_v - A_u\cdot A_w\cdot A_v \end{aligned} \tag{7}$$

The following theorem will formalize the intuitive notion that if a set of paths p_i with availabilities A_{p_i} have overlapping links that their total availability is less than when those paths would have been fully link disjoint.

Theorem 1: For given A_{p_i}, where $1 \leq i \leq k$, $A^k_{FD} \geq A^k_{PD}$.

Proof: A proof by mathematical induction:

When $k=2$, $A^2_{FD} = A_{p_1}+A_{p_2}-A_{p_1}\cdot A_{p_2}$, and $A^2_{PD} = A_{p_1}+A_{p_2}-A_{p_1}\circ A_{p_2}$. Since $A_{p_1}\cdot A_{p_2} \geq A_{p_1}\circ A_{p_2}$ according to Eq. (4) when $0 \leq A_{p_i} \leq 1$, the theorem is correct for $k=2$.

Assume when $k=m$ the theorem is correct:

$$A^m_{FD} = 1-\prod_{i=1}^{m}(1-A_{p_i}) \geq A^m_{PD} = 1-\coprod_{i=1}^{m}(1-A_{p_i}) \tag{8}$$

When $k=m+1$, $A^{m+1}_{FD} = 1-(1-A_{p_1})\cdot(1-A_{p_2})\cdots(1-A_{p_m})\cdot(1-A_{p_{m+1}})$ and $A^{m+1}_{PD} = 1-(1-A_{p_1})\circ(1-A_{p_2})\circ\circ\circ(1-A_{p_m})\circ(1-A_{p_{m+1}})$. According to Eq. (8), we have:

$$\begin{aligned} &(1-A_{p_1})\cdot(1-A_{p_2})\cdots(1-A_{p_m})\cdot(1-A_{p_{m+1}}) \\ \leq&(1-A_{p_1})\circ(1-A_{p_2})\circ\circ\circ(1-A_{p_m})\cdot(1-A_{p_{m+1}}) \end{aligned} \tag{9}$$

Since $(1-A_{p_m})\cdot(1-A_{p_{m+1}}) \leq (1-A_{p_m})\circ(1-A_{p_{m+1}})$, we have:

$$\begin{aligned} &(1-A_{p_1})\circ(1-A_{p_2})\circ\circ\circ(1-A_{p_m})\cdot(1-A_{p_{m+1}}) \\ \leq&(1-A_{p_1})\circ(1-A_{p_2})\circ\circ\circ(1-A_{p_m})\circ(1-A_{p_{m+1}}) \end{aligned} \tag{10}$$

By merging Eq. (9) and Eq. (10), we ascertain that $A^{m+1}_{FD} \geq A^{m+1}_{PD}$. ■

IV. ABPS PROBLEM AND COMPLEXITY

A. Problem Definition

Definition 1: Given a network represented by $G(\mathcal{N}, \mathcal{L})$ where $\mathcal{N}$ represents the set of N nodes and $\mathcal{L}$ denotes the set of L links, and each link l has its own availability value A_l. For a request represented by $r(s, t, \delta)$, where s and t denote the source and destination, respectively, and δ $(0 < \delta \leq 1)$ represents the availability requirement, establish a connection over at most k (partially) link-disjoint paths for which the availability is at least δ.

A variant, called the Availability-Based Backup Path Selection (ABBPS) problem, is defined as:

Definition 2: Given an existing primary path p from s to t and a requested availability δ, find at most $k-1$ paths that are fully or partially link-disjoint with p, such that the availability of these k paths is no less than δ.

B. Complexity Analysis

In this section, we study the complexity of the ABPS problem in generic networks. For the case $k = 1$, by taking the $-log$ of the link availabilities, the ABPS problem turns into a shortest path problem, which is polynomially solvable.

Theorem 2: The ABPS problem is NP-hard for $k \geq 2$.

Proof: The case for partially link-disjoint paths can be reduced to the case of fully link-disjoint paths by a transformation such as in Fig. 2. More specifically, if we assume that all links in Fig. 2, except for (s, s') and (t', t), have availability less than δ, then no link, except for (s, s') and (t', t), can be an unprotected link in the solution of the ABPS problem for the partially link disjoint case from s to t. Hence, solving the fully link-disjoint ABPS problem from s' to t' is equivalent to solving the partially link-disjoint ABPS problem from s to t. We therefore proceed to prove that the fully link disjoint variant for $k = 2$ is NP-hard. The proof for $k > 2$ follows analogously from the proof for $k = 2$.

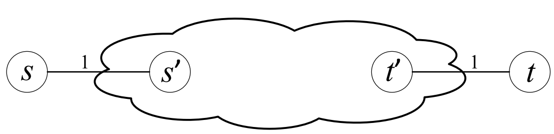


Fig. 2: Reduction of ABPS problem from partially link disjoint to fully link disjoint.

We first introduce the NP-hard 3SAT problem [18] and then reduce the ABPS problem to it. The 3SAT problem is defined as: There is a boolean formula $C_1 \wedge C_2 \wedge ... C_m$, where C_i denotes the i-th clause. Each clause contains 3 variables with an OR relation. The question is whether there is a truth assignment to the variables that simultaneously satisfies all m clauses. Given a 3SAT instance, the graph construction follows similarly to [1]. Assume there are n variables in the 3SAT instance. First, we create a lobe for each variable x_i, which is shown in Fig. 3, where q_i represents the number of occurrences of variable x_i in all the clauses. The availability value for each link is also shown in Fig. 3, where $0 < b < 1$. For each clause

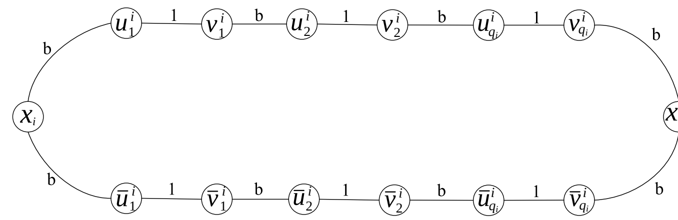


Fig. 3: A lobe for each x_i.

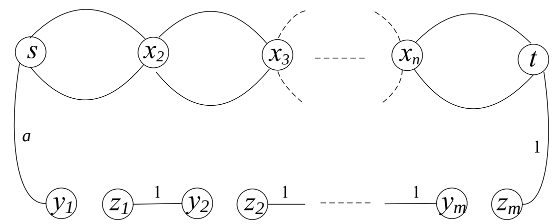


Fig. 4: Lobes for all clauses.

C_i two nodes y_i and z_i are created and a link connects z_i and y_{i+1} with availability of 1, where $0 < i < m$. We assume that $s = x_1$ and $t = x_{n+1}$. Moreover, we draw a link (s, y_1) with availability a and a link (z_m, t) with availability 1, where $0 < b < \frac{a}{2} < 1$. Fig. 4 depicts this process.

To relate the clause and variables in the constructed graph, we add the following links: (i) links (y_j, u_k^i) and (v_k^i, z_j) are added if the k-th occurrence of variable x_i exists in clause C_j; or (ii) links $(y_j, \overline{u}_k^i)$ and $(\overline{v}_k^i, z_j)$ are added if the k-th occurrence of variable x_i exists with a negation in the clause C_j. For instance, a network corresponding to 3SAT instance $(x_1 \wedge x_2 \wedge \overline{x}_4) \vee (\overline{x}_1 \wedge x_2 \wedge x_3) \vee (x_2 \wedge x_3 \wedge x_4) \vee (\overline{x}_1 \wedge \overline{x}_3 \wedge x_4)$ is shown in Fig. 5. Based on the constructed graph, which

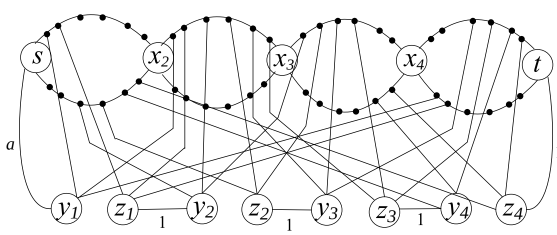


Fig. 5: Constructed graph that corresponds to $(x_1 \wedge x_2 \wedge \overline{x}_4) \vee (\overline{x}_1 \wedge x_2 \wedge x_3) \vee (x_2 \wedge x_3 \wedge x_4) \vee (\overline{x}_1 \wedge \overline{x}_3 \wedge x_4)$.

corresponds to a given 3SAT instance, we are asked to solve the ABPS problem for $k = 2$ and $\delta = a + b^q - ab^q$, where q is the sum of occurrences for each variable in all the clauses, i.e., $q = \sum_{i=1}^{n} q_i$. Because one shortest path can at most have availability a, which is less than δ, we have to find 2 link-disjoint paths. Next, we will prove that the fully link disjoint variant of the ABPS problem is NP-hard.

3SAT to ABPS: If there exists a truth assignment that satisfies all the clauses, then each clause j has (at least) one variable with true or (negated) false assignment to make this clause true. Therefore, an upper subpath $y_j - u_k^i - v_k^i - z_j - y_{j+1}$ or a lower subpath $y_j - \overline{u}_k^i - \overline{v}_k^i - z_j - y_{j+1}$ will be selected.

By concatenating these m subpaths with $s - y_1$ and $z_m - t$ we obtain one path (denoted by p_1) with availability a. Since each variable only has one truth assignment, p_1 cannot traverse both the upper subpath and lower subpath in the same lobe. Subsequently, we can get another fully link-disjoint path p_2: For each lobe i (corresponding to variable x_i), p_2 traverses the upper (lower) subpath with availability of b^{q_i} if p_1 goes through the link of lower (upper) subpath. The availability of p_2 is $b^q = b^{\sum_{i=1}^{n} q_i}$, therefore p_1 and p_2 together have availability of $a + b^q - ab^q$, which satisfies the requirement δ.

ABPS to 3SAT: If there are two fully link-disjoint paths from s to t with availability no less than $a + b^q - ab^q$, then one path must have availability a. To understand this, assume that none of the two paths has availability a; without loss of generality, we denote one path has availability of $a^c b^e$, where c can be either 0 or 1 indicating whether link (s, y_1) has been traversed, and $e > 0$ is the number of links that have availability b. Since there exists only one link with availability a, the other link-disjoint path has availability $a^{c'} b^f$, where c' is either 0 or 1 meaning whether link (s, y_1) has been traversed and $c' + c \leq 1$, and $f > 0$ is the number of links which have availability b. Hence, the availability of these two paths is $a^c b^e + a^{c'} b^f - a^{c+c'} b^{e+f} < b + b < a < \delta$, when $b < \frac{a}{2}$. Based on this analysis, there must exist one path p_1 from s to t with availability a, which goes through (s, y_1) and (z_m, t) and the other links with availability of 1. To satisfy the availability requirement, there must also exist another fully link-disjoint path p_2 from s to t with availability of no less than b^q. For each lobe, p_2 should traverse either the upper subpath or the lower subpath, otherwise p_1 and p_2 cannot be fully link disjoint. Therefore, p_2 will traverse the (entire) lower subpath if p_1 goes through link (u_k^i, v_k^i) in the upper subpath, and traverse the (entire) upper subpath if p_1 goes through link $(\overline{u}_k^i, \overline{v}_k^i)$ in the lower subpath for each lobe x_i. That is to say, p_1 cannot simultaneously traverse one link in the upper subpath and another link in the lower subpath for each same lobe. Consequently, p_1 either goes via an upper subpath $y_j - u_k^i - v_k^i - z_j - y_{j+1}$ to set variable x_i to true or via a lower subpath $y_j - \overline{u}_k^i - \overline{v}_k^i - z_j - y_{j+1}$ to set variable x_i to false for clause j, where $i = 1, 2, ..., n$ and $j = 1, 2, ..., m$. Hence, all the m clauses can be simultaneously satisfied. ∎

Theorem 3: The ABBPS problem is NP-hard for $k \geq 2$.

Proof: For $k \geq 3$, the ABBPS problem is equivalent to the ABPS problem for $k - 1$ fully or partially link-disjoint paths, and hence NP-hard. In Appendix B, we prove that the ABBPS problem is also NP-hard for $k = 2$. ∎

We proceed to study the approximability of the ABPS problem.

Theorem 4: The ABPS problem for $k \geq 2$ cannot be approximated to arbitrary degree in polynomial time, unless P=NP.

Proof: We can check in polynomial time whether a single path can accommodate the requested availability. Hence, the theorem is equivalent to: for a request $r(s, t, \delta)$ and any constant number $d > 1$, there is no polynomial-time algorithm that can find at least 2, but at most k, fully or partially link-disjoint paths from s to t with availability at least $\frac{\delta}{d}$. We prove the theorem for the fully link disjoint variant[3] of the ABPS problem for $k = 2$.

We will use a proof by contradiction and assume a polynomial-time approximation algorithm A exists for any $d > 1$. In the constructed graph based on the given 3SAT instance in Fig. 5 (also using the same notation and conditions), assume $\delta = a + b^q - ab^q$, so algorithm A can find two fully link-disjoint paths with availability at least $\frac{a+b^q-ab^q}{d}$. Next, we prove that when $0 < b < \frac{a}{2d}$, except for an exact solution, there exists no solution with availability no less than $\frac{a+b^q-ab^q}{d}$. If the exact solution is not achieved by algorithm A, according to our previous analysis, then one path must have availability of $a^c b^e$ and the other path has availability of $a^{c'} b^f$. Therefore, the availability of these two paths is equal to $a^c b^e + a^{c'} b^f - a^{c+c'} b^{e+f}$. For a given d, we have $a^c b^e + a^{c'} b^f - a^{c+c'} b^{e+f} < b + b = 2b < \frac{a}{d}$, when $0 < b < \frac{a}{2d}$ and $0 < a < 1$. Therefore, under $0 < b < \frac{a}{2d}$, except for an exact solution, any two fully link-disjoint paths cannot have availability less than $\frac{a+b^q-ab^q}{d}$. To fulfill the assumption, algorithm A has to find two link-disjoint paths with availability $a + b^q - ab^q$. In this context, the fully link disjoint variant of the ABPS problem for $k = 2$ can be solved exactly in polynomial time, which is a contradiction. ∎

V. Shared-Risk Link Groups

In this section, we assume two types of failures/availabilities, namely Shared-Risk Link Group (SRLG) failures and single link failures/availabilities. A Shared-Risk Link Group (SRLG) [16] reflects that a certain set/group of links in a network will fail simultaneously. For instance, in optical networks, several fibers may reside in the same duct and a cut of the duct would cut all fibers in it. One duct in this context corresponds to one distinct SRLG. If each link is a single member of an SRLG, then no SRLGs exist. Hence the ABPS problem in SRLG networks includes as a special case the ABPS problem in generic networks as discussed in the previous section. Each link can belong to one or more SRLGs, and the links in the same SRLG will simultaneously fail when the corresponding SRLG fails. The probability of this happening (or not) is the SRLG failure (availability) probability. We assume there are g SRLGs in the network $G(\mathcal{N}, \mathcal{L})$, and that the failure probability of the i-th SRLG (represented by $srlg_i$) is denoted by π_i, for $1 \leq i \leq g$. For a particular link $l \in \mathcal{L}$, we denote by SR^l the set of all SRLGs to which l belongs. Different from [2], where all SRLG events are assumed to be mutually exclusive, we assume that multiple SRLG events may occur simultaneously. The availability of a single path should incorporate the SRLG availabilities as well as the link availabilities. Consequently, the availability of path p can be calculated as:

$$\prod_{i; srlg_i \cap p \neq \emptyset} (1 - \pi_i) \prod_{l \in p} A_l \tag{11}$$

[3] The partially link disjoint variant follows analogously.

where $\prod_{i;srlg_i \cap p \neq \emptyset} (1-\pi_i)$ in Eq. (11) is the contribution of all the traversed SRLGs, while $\prod_{l \in p} A_l$ is the availability of path p under the condition that all its traversed SRLGs do not fail.

For example, in Fig. 6, suppose there are three SRLGs in the network with failure probabilities 0.1, 0.4 and 0.2, respectively, and all the links have availability 0.9. We calculate the availability of path $s-a-b-t$, which traverses 2 SRLGs ($srlg_1$ and $srlg_3$): The probability that both $srlg_1$ and $srlg_3$ do not fail is $(1-0.1) \times (1-0.2)$. Under this condition, all the links on path $s-a-b-t$ have availability 0.9 and therefore path $s-a-b-t$ has a total availability of $(1-0.1) \times (1-0.2) \times (0.9)^3 = 0.52488$.

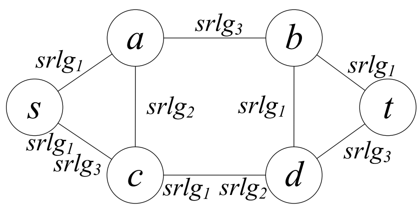


Fig. 6: Availability calculation in an SRLG network.

Next, we will prove that the single path variant of the ABPS problem in SRLG networks is NP-hard. To that end, we first introduce the Minimum Color Single-Path (MCSiP) problem, which is NP-hard [19]. Given a network $G(\mathcal{N}, \mathcal{L})$, and given the set of colors $C = \{c_1, c_2, ..., c_g\}$ where g is the total number of colors in G, and given the color $\{c_l\}$ of every link $l \in \mathcal{L}$, the Minimum Color Single-Path (MCSiP) problem is to find a path from source node s to destination node t that uses the least amount of colors.

Theorem 5: The ABPS problem is NP-hard in SRLG networks even for $k = 1$.

Proof: Assume we have a network where all the links have availability 1 when their SRLGs do not fail, and that there are g SRLGs with the same failure probability $\frac{1}{g}$. Hence, a path's availability is only determined by the number of SRLGs it traverses. If we denote one SRLG by one particular color, then the single-path ABPS problem in SRLG networks can be reduced to the MSCiP problem. ∎

VI. Multi-layer networks

In multi-layer (e.g., IP-over-WDM) networks or overlay networks, the abstract links in the logical layer are mapped to different physical links in the physical layer. In this context, two or more abstract links that contain the same physical links may have correlated availabilities or failure probabilities. Moreover, usually only the links in the logical layer are known in multi-layer networks.

Let us first consider the example of multi-layer networks shown in Fig. 7. In Fig. 7(a), the availability is labeled on each link in the physical layer, and the links in the logical layer are mapped to the links in the physical layer with the greatest availability. Suppose we want to find a most reliable unprotected path from s to t in Fig. 7(a). Since we are only aware of the links in the logical layer, we find that the most reliable path's availability is $0.72 \cdot 0.4 = 0.288$. However, the optimal solution is path s-b-t with availability 0.45 in the physical layer. The reason is that (s, a) and (a, t) in the logical layer share the same link (s, b), which leads to a lower availability value. Fig. 7(b) shows a similar example with Fig. 7(a), except that each link in the physical layer has one additional wavelength number. In the absence of wavelength conversion, it is required that the lightpath occupies the same wavelength on all links it traverses, which is referred to as the wavelength-continuity constraint in WDM-enabled networks. Now, suppose we want to find the most reliable lightpath from s to t, that obeys the wavelength-continuity constraint. Clearly, if we are only aware of the links in the logical layer, the result is path s-a-t with availability $0.8 \cdot 0.45 = 0.36$. However, since this path is mapped to the path s-a-b-t in the physical layer, it violates the wavelength-continuity constraint. The optimal solution is path s-a-t in the physical layer via wavelength λ_1. Its availability is $0.8 \cdot 0.3 = 0.24$.

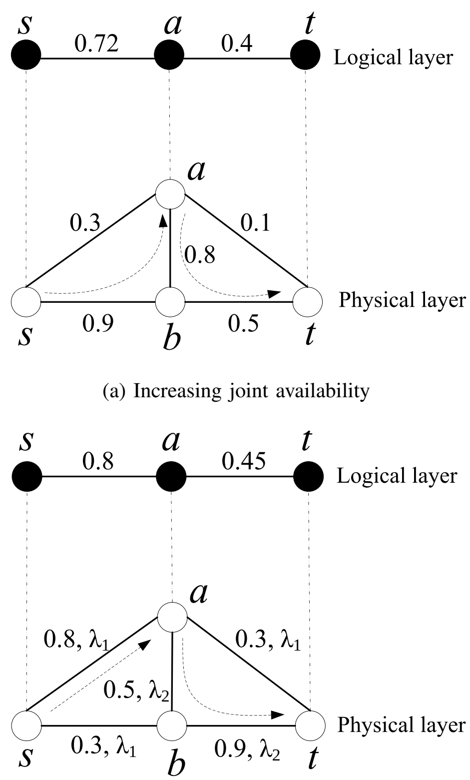


Fig. 7: A multi-layer network example.

For any two links l and m in the logical layer of a multi-layer network, we denote their availabilities by A_l and A_m, respectively. Let us use $\odot$ as the actual joint availability value of these two links, and then we derive that $A_l \odot A_m$ can be greater than $A_l \cdot A_m$ (Fig. 7(a)), or less than $A_l \cdot A_m$ (Fig. 7(b)), or equal to $A_l \cdot A_m$. For the latter case, we say that l and m are uncorrelated, otherwise we say that l and m are correlated: if $A_l \odot A_m > A_l \cdot A_m$, l and m are "increasing correlated," else l and m are "decreasing correlated." Analogously, the operator

$\odot$ can be used for more than two links. Next, we will prove the NP-hardness of the ABPS problem for $k = 1$ in multi-layer networks.

Theorem 6: The ABPS problem is NP-hard in multi-layer networks even for $k = 1$.

Proof: When all the links are uncorrelated in multi-layer networks, the ABPS with $k = 1$ problem is solvable polynomial time. SRLG networks can be regarded as a special case of increasing correlation in multi-layer networks, since the links that share at least one common SRLG group (denote this link set by $\mathcal{L}'$) will have a greater availability than $\prod_{l \in \mathcal{L}'} \left(\prod_{i; srlg_i \cap l \neq \emptyset} (1 - \pi_i) \cdot A_l \right)$. Since the single-path ABPS problem in SRLG networks is NP-hard, as we proved in Theorem 5, the single-path ABPS problem in multi-layer networks is also NP-hard. ■

VII. Heuristic and Exact Algorithms

A. Heuristic Algorithm

Algorithm 1 $MMA(G, s, t, \delta, k, I)$

1: Find one shortest path p_1, return it if the availability requirement is satisfied, otherwise go to Step 2.
2: $ps \leftarrow 1$, $H \leftarrow p_1$, $P \leftarrow H$, $P_b \leftarrow \emptyset$ and $Q \leftarrow \emptyset$
3: **While** $ps \leq k$
4:   $P \leftarrow H$
5:   **For** each path $ap \in P$
6:     $P_b \leftarrow P$ and $counter \leftarrow 0$
7:     **While** $counter \leq I$ **do**
8:       Randomly remove one link $(u, v) \in ap$ and find one shortest path $\psi_{u \to v}$ from u to v.
9:       **If** it succeeds **then**
10:         Replace (u, v) with $\psi_{u \to v}$ in ap, denote it as ap'
11:         P_b. Remove(ap), P_b. Add(ap'), $ap \leftarrow ap'$
12:         Find another link-disjoint path p_2 with P_b.
13:         Return $\{p_2\} \cup P_b$ if δ is met.
14:       **For** each link $(u, v) \in ap'$
15:         **If** its availability is at least δ **then**
16:           Q. Add((u, v))
17:       **while** $(Q \neq \emptyset)$ **do**
18:         $(u, v) \leftarrow$ Extract-Min(Q)
19:         Find a path p_3 which shares (u, v) with ap'.
20:           **If** $(p_3 \notin P_b)$ and $\{p_3\} \cup P_b$ satisfy δ **then**
21:             Return $\{p_3\} \cup P_b$
22:           **else** $H \leftarrow Max\_Availability\{H, \{p_3\} \cup P_b\}$
23:       $counter \leftarrow counter + 1$.
24:   $ps \leftarrow ps + 1$

Our heuristic, called Min-Mins Algorithm (MMA) to solve the ABPS problem in generic networks, SRLG networks and multi-layer networks, is presented in Algorithm 1. Since we want to use as least (and no more than k) link-disjoint paths to satisfy the requested availability, we gradually increase the number of paths.

The pseudo code to solve the ABPS problem in multi-layer networks is similar to the one for generic networks, except for the path availability calculation. Also, the pseudo code to solve the ABPS problem in SRLG networks is similar to the one in generic networks, and we will specify the differences later. In what follows, we explain each step of the heuristic algorithm. We assign link $l \in \mathcal{L}$ with the weight of $-log(A_l)$ ($-log(\prod_{i \in SR^l}(1 - \pi_i) \cdot A_l)$ for SRLG networks) in MMA. If a shortest path (represented by p_1) in Step 1 fails to satisfy the availability requirement, we keep it as the initial path flow. In Step 2, we use ps to record the number of already found link-disjoint paths. Initially ps is set to 1. H stores the already found ps link-disjoint paths, and it is initially assigned p_1. While ps is no greater than k, Steps 3-24 continue finding a solution. In Step 4, we assign to P the already found paths H. Based on P, from Step 5 to Step 23, we each time select one path ap from path set P. We also use a variable, denoted by $counter$ in Algorithm 1, to record the number of iterations. Initially, $counter$ is set to 0. As long as the number of iterations is less than an input value I, Steps 7-23 proceed finding a solution based on ap and path set P_b. The (sub)path from u to v found by the algorithm is denoted by $\psi_{u \to v}$. In Step 8, we randomly remove one link (u, v) from ap, and we apply a shortest path algorithm from u to v to obtain a path $\psi_{u \to v}$. By concatenating subpath $\psi_{u \to v}$ and the links of path ap except for (u, v), we obtain a new path ap'. Further, by substituting ap with ap' in P_b, we have a new path set P_b. After that, the algorithm tries to find P_b's fully link-disjoint path in Step 12. When solving the ABPS problem in SRLG networks, since each SRLG only contributes once to the path availability calculation, the link l's weight is set to $-log(\prod_{i \in \{SR^l \setminus SR^c\}}(1 - \pi_i) \cdot A_l)$ before running a shortest path algorithm in Step 12 (also the same for Step 19), where SR^c are the common traversed SRLGs between link l and path set P_b. If it fails to find p_2 or $\{p_2\} \cup P_b$ cannot satisfy the availability requirement, the algorithm tries to find a path which is partially link disjoint with ap' (in Steps 13-22). The general idea is that we first use a queue Q to store the links in ap' whose availability is no less than δ in Steps 14-16. After that, as long as Q is not empty in Steps 17-22, each time the link with the greatest availability in Q is extracted as the unprotected link (represented by (u, v)), and then we remove all the links traversed by ap' except for (u, v). Subsequently, we find one shortest path $\psi_{s \to u}$ from s to u (if it exists), and find another shortest path $\psi_{v \to t}$ from v to t (if it exists). By concatenating $\psi_{s \to u}$, (u, v) and $\psi_{v \to t}$, we can get a new path p_3, which is partially link disjoint with ap'. If a and b denote different sets of $k > 1$ link-disjoint paths, the function Max_Availability(a, b) in Step 22 returns the one with greater availability.

The time complexity of MMA can be computed as follows. Step 1 has a time complexity of $O(N \log N + L)$. From Step 3 to Step 24, there are at most $O(I) + O(2I) + \cdots + O(kI) = O(k^2 I)$ iterations before the algorithm terminates. Steps 14-16 have a time complexity of $O(N)$ since a path contains at most $N - 1$ links and therefore Steps 17-22 consume

$O(N(N \log N + L))$ time. Finally, the whole time complexity of MMA is $O(k^2 IN(N \log N + L))$.

B. Exact INLP Formulation

In this subsection, we present an exact Integer Non-Linear Program (INLP) to solve the ABPS problem in generic, SRLG and multi-layer networks. We first solve the ABPS problem in generic networks and start by explaining the required notation and variables.

INLP notation:

$r(s, t, \delta)$: Traffic request, with source s, destination t and requested availability δ.

$A_{i,j}$: Availability of link (i, j).

g : The total number of SRLGs.

$\pi^m_{i,j}$: The failure probability of the m-th SRLG if link (i, j) belongs to it, otherwise it is 0.

INLP variable:

$P^{r,u}_{i,j}$: Boolean variable equal to 1 if link (i, j) is traversed by path u $(1 \leq u \leq k)$ for request r; 0 otherwise.

Flow conservation constraints:

$$\sum_{(i,j)\in\mathcal{L}} P^{r,u}_{i,j} - \sum_{(j,i)\in\mathcal{L}} P^{r,u}_{j,i} = \begin{cases} 1, & i = s \\ -1, & i = t \\ 0, & \text{otherwise} \end{cases} \quad (12)$$
$$\forall i \in \mathcal{N} \quad 1 \leq u \leq k$$

Availability constraint:

$$\sum_{u=1}^{k} \prod_{(i,j)\in\mathcal{L}} \left(1 - P^{r,u}_{i,j} + P^{r}_{i,j} A_{i,j}\right) -$$
$$\sum_{1\leq u<v\leq k} \prod_{(i,j)\in\mathcal{L}} \min\left(1 - P^{r,u}_{i,j} + P^{r,u}_{i,j} A_{i,j}, 1 - P^{r,v}_{i,j} + P^{r,v}_{i,j} A_{i,j}\right)$$
$$+ \cdots + (-1)^{k-1} \left(\prod_{(i,j)\in\mathcal{L}} \min_{1\leq u\leq k} (1 - P^{r,u}_{i,j} + P^{r,u}_{i,j} A_{i,j})\right) \geq \delta \quad (13)$$

When both the flow conservation constraint (Eq. (12)) and the availability constraint (Eq. (13)) are satisfied, an optimal solution is found by the INLP, otherwise there is no solution. There is no objective (needed) in the proposed INLP, but one could include the objective of minimizing the number of paths (or links) used. Eq. (12) accounts for the flow conservation for each of the at most k paths. For a particular u^{th} path $(1 \leq u \leq k)$, it ensures that (i) for the source node s of request r, the outgoing traffic for each request is 1; (ii) for the destination node t of request r, the incoming traffic is 1; and (iii) for an intermediate node which is neither source nor destination, its incoming traffic is equal to the outgoing traffic. Eq. (13) ensures that either the found single unprotected path or the (partially) link-disjoint paths should have availability no less than δ, according to the availability calculation of k link-disjoint paths in Eqs. (3) and (5). Since the overlapped link's availability in the partially link-disjoint calculation according to Eq. (5) can only be counted once, we take the minimum value of the variables $P^{r,u}_{i,j}$ for each link and then take the product over all the links for (partially) link-disjoint paths. We also note that Eq. (13) can simultaneously calculate the availability of the fully link disjoint variant, partially link disjoint variant and the unprotected variant. For instance when $k = 2$, Eq. (13) becomes:

$$\prod_{(i,j)\in\mathcal{L}} (1 - P^{r,1}_{i,j} + P^{r,1}_{i,j} A_{i,j}) + \prod_{(i,j)\in\mathcal{L}} (1 - P^{r,2}_{i,j} + P^{r,2}_{i,j} A_{i,j}) -$$
$$\prod_{(i,j)\in\mathcal{L}} min(1 - P^{r,1}_{i,j} + P^{r,1}_{i,j} A_{i,j}, 1 - P^{r,2}_{i,j} + P^{r,2}_{i,j} A_{i,j}) \geq \delta \quad (14)$$

When $P^{r,1}_{i,j} = P^{r,2}_{i,j}$ for all $(i, j) \in \mathcal{L}$, Eq. (14) is equal to

$$\prod_{(i,j)\in\mathcal{L}} (1 - P^{r,1}_{i,j} + P^{r,1}_{i,j} A_{i,j}) \geq \delta$$

or

$$\prod_{(i,j)\in\mathcal{L}} (1 - P^{r,2}_{i,j} + P^{r,2}_{i,j} A_{i,j}) \geq \delta$$

which is the availability constraint for a single unprotected path.

To solve the ABPS problem in SRLG networks, we need to slightly modify Eq. (13) (and keep flow conversation constraints Eq. (12) the same) by using $\prod_{1\leq m\leq g} \min_{u=1}^{k} \left(1 - \min_{(i,j)\in\mathcal{L}} (1 - P^{r,u}_{i,j} + P^{r,u}_{i,j} \pi^m_{i,j})\right)$ to multiply the left side of Eq. (13), which is the non-failure probability of the SRLGs which at most k link-disjoint paths have traversed.

In multi-layer networks, the availability of a subset of links may not be equal to the product of their availabilities. Therefore, we need one more function $f(\overrightarrow{L})$, which can return the joint availability of a subset of links $\overrightarrow{L}$ in multi-layer networks. The parameter of this function is a $0/1$ link vector which contains L elements, where 1 denotes the link is present to be calculated and 0 means it is not. Consequently, to solve the ABPS problem in multilayer networks exactly, we could replace the operator $\prod$ with $f()$ in Eq. (13), and keep all the other notation and constraints the same as for the generic networks case.

VIII. Simulation Results

A. Simulation Setup

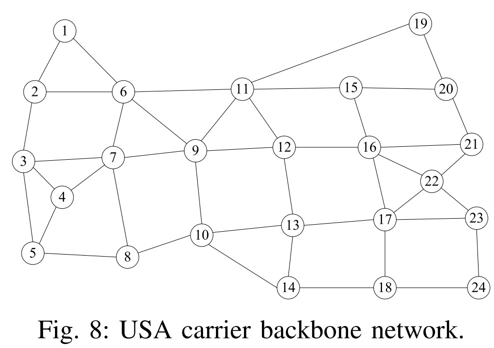


Fig. 8: USA carrier backbone network.

We conduct simulations on two networks, one is USANet, displayed in Fig. 8, which is a realistic carrier backbone network consisting of 24 nodes and 43 links, and the other is GÉANT, shown in Fig. 9, which is the pan-European communications infrastructure serving Europe's research and

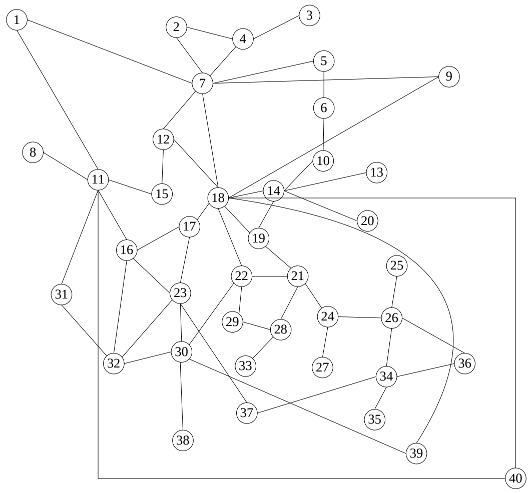


Fig. 9: GÉANT pan-European research network.

education community consisting of 40 nodes and 63 links. The simulation deals with the ABPS problem in generic, SRLG and multi-layer networks. For generic networks, we assume the availability of fiber links is distributed among the set $\{0.99$, 0.999, $0.9999\}$, with a proportion of 1:1:2. Based on the same link availabilities, in SRLG networks we assume that there are in total 5 SRLG events with the failure probabilities 0.001, 0.002, 0.003, 0.004 and 0.005, respectively. Each link has randomly been assigned to at most 3 SRLG events. Based on the same (individual) link availabilities as for generic networks, in multi-layer networks, $\frac{1}{3}$ of links are increasing correlated, and it follows that $A_l \odot A_m \odot \cdots \odot A_n = \max(A_l, A_m, \cdots, A_n)$; $\frac{1}{3}$ of links are decreasing correlated, and it follows that $A_l \odot A_m \odot \cdots \odot A_n = (A_l \cdot A_m \cdot \cdots \cdot A_n)^2$; and the other $\frac{1}{3}$ of links are uncorrelated. For all these three networks, since we want to compare the ability of finding paths for the algorithms, the capacity is set to infinity. We vary the number of traffic requests from 100 to 1000. The source and destination of each request are randomly selected, and each request has infinite holding time. The requested availability includes two cases: (i) general availability requirement case: the availability is randomly distributed among the set $\{0.98$, 0.99, 0.995, 0.997, $0.999\}$; (ii) high availability requirement case: the availability is randomly distributed among the set $\{0.9995$, 0.9996, 0.9997, 0.9998, $0.9999\}$, by which we want to challenge the algorithm to find feasible paths under more difficult conditions. Considering the practical time complexity and the existing proposed algorithms that only focus on finding two link-disjoint paths, we choose $k = 2$. We set I in MMA to be $\lceil logN \rceil$ in these two networks (5 in USANet and 6 in GÉANT, respectively). Under the same weight allocation with our algorithm, we compare the proposed heuristic MMA and exact INLP with two heuristics: Two-step Reliability Algorithm (TRA) and Maximal-Reliability Algorithm (MRA), which are proposed in [1]. TRA first calculates a shortest path, and then calculates (if it exists) another shortest path after removing the links traversed by the first path. MRA applies Suurballe's algorithm [20] to calculate a pair of two link-disjoint paths that have minimum weight. Both algorithms first apply a shortest path algorithm to check whether an unprotected path solution exits. The simulation is run on a desktop PC with 3.00 GHz CPU and 4 GB memory. We use IBM ILOG CPLEX 12.6 to implement the proposed INLP and C# to implement the heuristic algorithms.

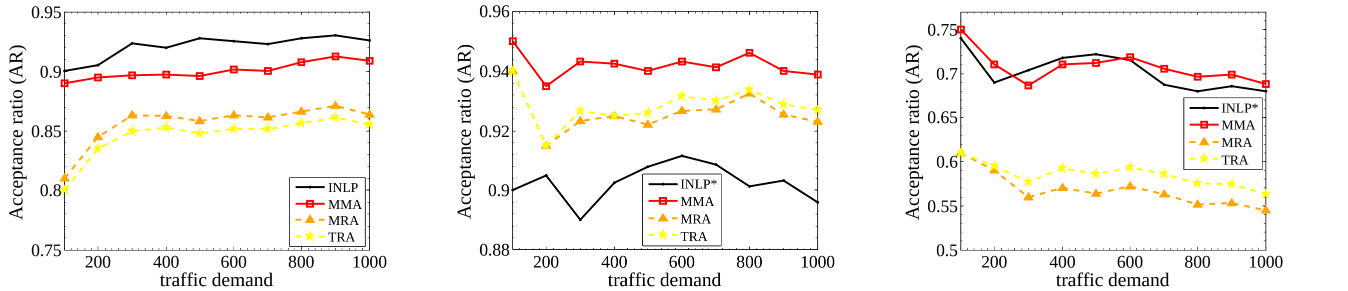
B. Results

We first evaluate the performance of the algorithms in terms of Acceptance Ratio (AR) in generic networks. Acceptance ratio (AR) is defined as the percentage of the number of accepted requests over all the requests. We first analyze the general availability requirement case: In USANet, all the algorithms achieved an AR of 1. We therefore omit the figure of the general availability performance for USANet. However, this is not the case for the GÉANT topology. From Fig. 10(a), we can see that the performance of all algorithms is under 0.95. Since GÉANT is not as well connected as USANet is, some nodes in GÉANT only have degree one (e.g., nodes 3, 8, etc.), if a one-degree node becomes the source or the destination of a certain request, the request can only be served by partial protection (or a single unprotected path). In this context, a feasible path may not exist in GÉANT, which will result in blocking. In terms of performance, the INLP achieves the highest AR. On the other hand, MMA shows a higher AR than the other two heuristics TRA and MRA (Fig. 10(a)).

For the high availability requirement scenario (shown in Figs. 10(b) and 10(c)), as expected, the AR of all these algorithms is lower than in the general availability requirement case. In this scenario, the INLP requires more time to find a solution, especially when a solution does not exist. In order to let the INLP return the result in a reasonable time, we set the time limit for it to serve one request to 50 minutes. Due to this reason, we can see that INLP has the lowest AR in USANet and often second highest AR in GÉANT. Meanwhile, MMA still has the highest AR in most of the cases.

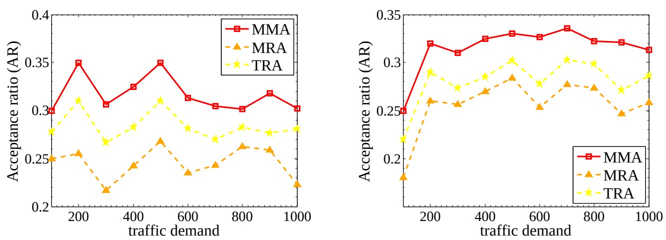
The time limit for the INLP is even more constraining in the case of SRLG networks, leading to a very poor performance for SRLG networks. We have therefore omitted the results of the INLP in SRLG networks. Since the optimal solution rarely exists in the high availability requirement case, we only provide the simulation results for the heuristic algorithms in the general availability requirement case. Moreover, to have a fair comparison, we compare our algorithms with MRA and a modified TRA [2], which is a heuristic routing algorithm proposed for probabilistic SRLG networks. Its main idea is that after finding the first shortest path, the remaining link weights should be adjusted (We slightly change its link weight adjustment to be the same with the Step 12 of MMA for a fairer comparison), and then to find another link-disjoint shortest path. Fig. 11 shows that the proposed heuristic algorithm MMA still achieves higher AR than these two algorithms.

Similar to SRLG networks, the exact INLP in multi-layer networks is very time consuming. We therefore omit the results of the INLP in multi-layer networks. Fig. 12 provides the



(a) GÉANT (b) USANet (c) GÉANT

Fig. 10: AR of four algorithms in generic networks: (a) general availability requirement. (USANet has been omitted since all 4 algorithms always achieved an AR of 1.); (b)-(c) high availability requirement, * max 50 mins per request.



(a) USANet (b) GÉANT

Fig. 11: AR of the heuristic algorithms in SRLG networks for general availability requirement.

results for all three heuristics in the two networks for both general and high availability requirement scenarios. It can be seen that MMA achieves the highest AR compared to the other two heuristics.

TABLE I: Running times per request for four algorithms (ms).

Networks	INLP	MMA	MRA	TRA
USA Generic (General δ)	10190	0.187	0.128	0.127
GÉANT Generic (General δ)	29896	0.558	0.143	0.142
USA Generic (High δ)	79764	0.224	0.147	0.146
GÉANT Generic (High δ)	135181	0.679	0.162	0.160
USA SRLG (General δ)	$> 3.6 \cdot 10^7$	0.461	0.136	0.161
GÉANT SRLG (General δ)	$> 3.6 \cdot 10^7$	0.663	0.167	0.196
USA Multi-layer (General δ)	$> 3.6 \cdot 10^7$	0.857	0.135	0.141
GÉANT Multi-layer (General δ)	$> 3.6 \cdot 10^7$	1.378	0.175	0.181
USA Multi-layer (High δ)	$> 3.6 \cdot 10^7$	1.136	0.157	0.162
GÉANT Multi-layer (High δ)	$> 3.6 \cdot 10^7$	1.679	0.262	0.223

Finally, in Table I, we present the (average) running times per request for these four algorithms in generic, SRLG and multi-layer networks. It shows that the INLP is significantly more time consuming than the three polynomial-time heuristics. On the other hand, MMA has only a slightly higher running time than MRA and TRA, but it pays off by having a higher AR as shown in Figs. 10-12. Another observation is that, for the same algorithm in the same network, the running time is higher for the high availability requirement case than in the general availability requirement case.

IX. Network Vulnerability Assessment

A. Problem Definition and Complexity Analysis

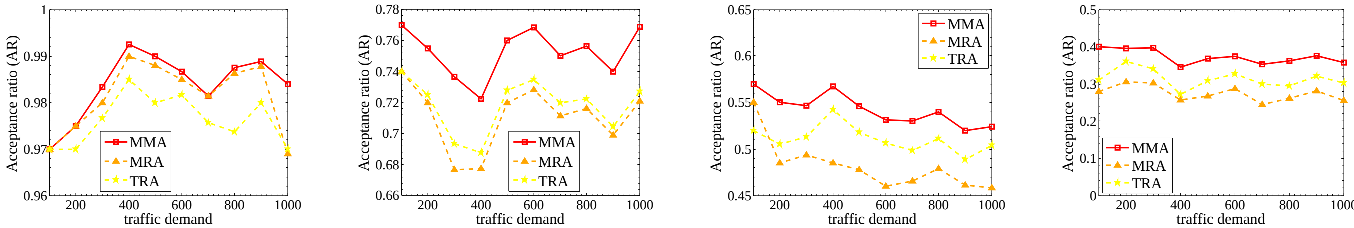
Finding the most vulnerable part of a network as well as the previously considered problem of availability-based path selection are both important elements for network robustness. In this section, we study the (s,t) Network Vulnerability Assessment (NVA) problem. That is, find one or a set of (equal-weight) network cuts whose failure probability of the links in the cut is highest. A network cut refers to a set of links, whose removal will result in the disconnection of the network. Formally, the NVA problem can be defined as follows:

Definition 3: The (s,t) Network Vulnerability Assessment (NVA) problem: Given is a network $G(\mathcal{N},\mathcal{L})$, and each link $l \in \mathcal{L}$ is associated with a failure probability $f_l = 1 - A_l$. Given a source s and a target t, find an $s-t$ cut $\mathcal{C}$ for which $\prod_{l \in C} f_l$ is maximized. In case there are multiple cuts of highest weight all of them should be returned.

When the node pair (s,t) is not specified, we denote this problem as the NVA problem, which can be solved by solving the (s,t) NVA problem at most $N-1$ times. Therefore, these two problems share the same hardness.

We use $-\log(f_l)$ for the weight of link l in the network. In generic networks, the NVA problem can be solved by finding all the min-cuts in $O(L^2N + N^2L)$ time according to [21]. On the other hand, we will prove that the NVA problem in SRLG networks is NP-hard. Recall that in SRLG networks, introduced in Section V, two types of failures/availabilities should be incorporated, namely Shared-Risk Link Group (SRLG) failures and single link failures/availabilities, therefore the probability that all the links belonging to path p fail simultaneously (denoted by $F(p)$) can be calculated as:

$$\left[1 - \prod_{i;srlg_i \cap p \neq \emptyset} (1-\pi_i)\right] + \prod_{l \in p} f_l - \left[1 - \prod_{i;srlg_i \cap p \neq \emptyset} (1-\pi_i)\right] \cdot \prod_{l \in p} f_l \tag{15}$$



(a) USANet (b) GÉANT (c) USANet (d) GÉANT

Fig. 12: AR of heuristics in multi-layer networks: (a)-(b) general availability requirement; (c)-(d) high availability requirement.

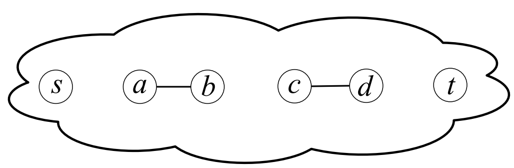
Please note that Eq. (15) is different from the failure probability of a path in SRLG networks: 1−Eq. (11).

Before we prove the NP-hardness of the NVA problem, let us first study the Maximum Color Path Selection (MCPS) problem. Contrary to the MCSiP problem, the MCPS problem is to find a single path such that it uses the largest amount of colors.

Theorem 7: The Maximum Color Path Selection (MCPS) problem is NP-hard.

Proof: We distinguish two cases, namely, (1) all the links have exclusive colors, i.e., there does not exist any color that is shared/overlapped by two or more links, and (2) two or more links may contain the same color(s).

Case (1): The MCPS problem is equivalent to the NP-hard Longest Path problem [18], which is to find a path from s to t such that its weight is maximized.



Links (a,b) and (c,d) together contain totally g distinct colors.

Fig. 13: Reduction of the MCPS problem to the Disjoint Connecting Paths problem.

Case (2): We first introduce the Disjoint Connecting Paths problem [18]. Given a network $G(\mathcal{N}, \mathcal{L})$, and a collection of disjoint node pairs (s_1, t_1), (s_2, t_2), ..., (s_z, t_z), does G contain z mutually link-disjoint paths, one connecting s_i and t_i for each i, $1 \leq i \leq z$? This problem is proved NP-hard when $z \geq 3$, and we reduce the MCPS problem to it. In Fig. 13, assume that there are in total g colors and links (a, b) and (c, d) share $0 < x < g$ common colors, but they together contain g distinct colors. Except for these two links, the other links are assigned $0 < y < g$ colors, but there does not exist two or more links containing g distinct colors. In this context, finding a path from s to t with the largest amount of colors is equivalent to finding three mutually link-disjoint paths between three node pairs (s, a), (b, c) and (d, t). ■

Theorem 8: The Network Vulnerability Assessment (NVA) problem in SRLG Networks is NP-hard.

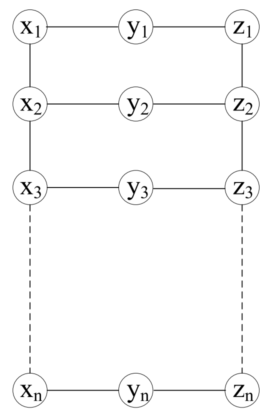


Fig. 14: Example network.

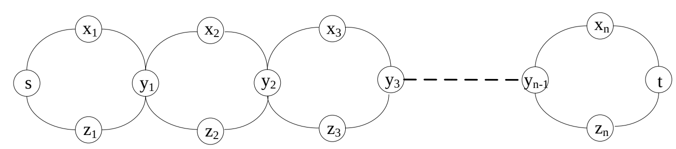


Fig. 15: Reduction of the (s,t) NVA problem in SRLG networks to the MCPS problem.

Proof: It is equivalent to prove that the (s,t) NVA problem is NP-hard.

In Fig. 14, assume all link availabilities are 1 and all links have non-zero SRLG failure probabilities, except for links (x_i, x_{i+1}) and (z_i, z_{i+1}) which have 0 SRLGs, where $1 \leq i \leq n-1$. Assume there are g SRLG events in total, and each SRLG event occurs with a probability of $\frac{1}{g}$. In this context, for a path p without links (x_i, x_{i+1}) and (z_i, z_{i+1}), according to Eq. (15), $F(p)$ can be calculated as:

$$1 - (1 - \frac{1}{g})^m \tag{16}$$

where m is the total number of distinct SRLGs traversed by p. Therefore, to maximize Eq. (16) one needs to maximize m, i.e., to find a path with the greatest probability that all its traversed links fail simultaneously. This is equivalent to finding a path having the largest number of distinct SRLGs. We want to solve the (x_1,z_1) NVA problem. Based on Fig. 14, we first derive Fig. 15 with the same nodes except that we add one more node s. We regard $s = y_0$, and $t = y_n$. The link weight in Fig. 15 is set as follows: (y_{i-1}, x_i) and (y_{i-1}, z_i) have 0 SRLGs, while (x_i, z_i) and (z_i, y_i) have the same SRLGs as in Fig. 14, where $1 \leq i \leq n$. In Fig. 15, we are asked to solve the MCPS problem from the source s to the destination t.

Since we want to find a cut that separates x_1 and z_1, any cuts in the form of (x_i, y_i) and (y_i, z_i), where $1 \leq i \leq n$ are not part of the optimal solution. Moreover, considering the link in the form of (x_j, x_{j+1}) or (y_j, y_{j+1}) has 0 SRLGs and single link availability 1, which means its availability is 1 or failure probability is 0, it cannot lead to the optimal solution as well. Based on the above observations, any feasible cut C should contain one link either (x_i, y_i) or (y_i, z_i), for all $1 \leq i \leq n$. We prove in the following that the (s,t) NVA problem in Fig. 14 can be reduced to the MCPS problem in Fig. 15 in polynomial time, where, for simplicity, we assume only one optimal cut exists in the (s,t) NVA problem.

(s,t) NVA to MCPS: An optimal solution of the (s,t) NVA problem should be composed of either (x_i, y_i) or (y_i, z_i), where $1 \leq i \leq n$. Let C_{NVA} reflect the set of links in the optimal solution. Because C_{NVA} has the largest amount of distinct SRLGs, C_{NVA} together with (s, x_1) or (s, z_1) forms a path from s to t with the maximum number of SRLGs.

MCPS to (s,t) NVA: Let R_{MCPS} denote the set of links in the optimal solution of the MCPS problem. Because R_{MCPS} has the largest amount of SRLGs, let $C_{NVA} = R_{MCPS} \backslash \{(y_i, x_{i+1}), (y_i, z_{i+1})\}$. Considering that the links (y_i, x_{i+1}) and (y_i, z_{i+1}) have 0 SRLGs, C_{NVA} also has the largest amount of SRLGs. Therefore solving the MCPS problem yields a solution to the (s,t) NVA problem. ■

Theorem 9: The NVA problem in multi-layer networks is NP-hard.

Proof: For any two links l and m in a multi-layer network, we have that $A_l \odot A_m = (1 - f_l) \odot (1 - f_m) = 1 - f_m - f_l + f_l \odot f_m$. Similar to Theorem 6, we can also prove that it is NP-hard to find one path from the source to the destination in multi-layer networks such that the probability that all its links fail is maximized. Subsequently, we can prove that the NVA problem in multi-layer networks is NP-hard, which follows analogously with Theorem 8. We therefore omit the details. ■

X. Conclusion

The availability of a connection represents how reliably a connection can carry data from a source to a destination. In this paper, we have first studied the Availability-Based Path Selection (ABPS) problem, which is to establish a connection over at most k (fully or partially) link-disjoint paths, such that the total availability is no less than δ ($0 < \delta \leq 1$). We have proved that, in general, the ABPS problem is NP-hard and cannot be approximated in polynomial time for $k \geq 2$, unless P=NP. We have further proved that in SRLG networks and multi-layer networks, even the single-path ($k = 1$) variant of the ABPS problem is NP-hard.

We have proposed a polynomial-time heuristic algorithm and an exact INLP to solve the ABPS problem in generic networks, SRLG networks and multi-layer networks. Via simulations, we have found that our heuristic algorithm outperforms two existing algorithms in terms of acceptance ratio with only slightly higher running time. On the other hand, the running time of the exact INLP is significantly larger (by several orders of magnitude) than the running time of the heuristic algorithms.

Finally, we have proved that the Network Vulnerability Assessment (NVA) problem, which is to find a cut of the network for which the failure probability of all its links is highest, is solvable in polynomial time in generic networks, but NP-hard to solve in SRLG networks and multi-layer networks.

References

[1] Q. She, X. Huang, and J. Jue, "How reliable can two-path protection be?" *IEEE/ACM Transactions on Networking*, vol. 18, no. 3, pp. 922–933, 2010.

[2] H.-W. Lee, E. Modiano, and K. Lee, "Diverse routing in networks with probabilistic failures," *IEEE/ACM Transactions on Networking*, vol. 18, no. 6, pp. 1895–1907, 2010.

[3] J. Zhang, K. Zhu, H. Zang, N. Matloff, and B. Mukherjee, "Availability-aware provisioning strategies for differentiated protection services in wavelength-convertible wdm mesh networks," *IEEE/ACM Transactions on Networking*, vol. 15, no. 5, pp. 1177–1190, 2007.

[4] M. Tornatore, G. Maier, and A. Pattavina, "Availability design of optical transport networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 8, pp. 1520–1532, 2005.

[5] L. Song, J. Zhang, and B. Mukherjee, "Dynamic provisioning with availability guarantee for differentiated services in survivable mesh networks," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 3, pp. 35–43, 2007.

[6] H. Luo, L. Li, and H. Yu, "Routing connections with differentiated reliability requirements in wdm mesh networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp. 253–266, 2009.

[7] J. S. Provan and M. O. Ball, "The complexity of counting cuts and of computing the probability that a graph is connected," *SIAM Journal on Computing*, vol. 12, no. 4, pp. 777–788, 1983.

[8] L. G. Valiant, "The complexity of enumeration and reliability problems," *SIAM Journal on Computing*, vol. 8, no. 3, pp. 410–421, 1979.

[9] D. R. Karger, "A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem," *SIAM review*, vol. 43, no. 3, pp. 499–522, 2001.

[10] W. Zou, M. Janić, R. Kooij, and F. Kuipers, "On the availability of networks," in *Proc. of BroadBand Europe*, Belgium, December, 2007.

[11] M. Tornatore, M. Carcagni, and A. Pattavina, "Availability formulations for segment protection," *IEEE Transactions on Communications*, vol. 58, no. 4, pp. 1031–1035, 2010.

[12] D. A. A. Mello, D. Schupke, and H. Waldman, "A matrix-based analytical approach to connection unavailability estimation in shared backup path protection," *IEEE Communications Letters*, vol. 9, no. 9, pp. 844–846, 2005.

[13] J. Q. Hu, "Diverse routing in optical mesh networks," *IEEE Transactions on Communications*, vol. 51, no. 3, pp. 489–494, 2003.

[14] D. Xu, Y. Xiong, C. Qiao, and G. Li, "Trap avoidance and protection schemes in networks with shared risk link groups," *Journal of Lightwave Technology*, vol. 21, no. 11, pp. 2683–2693, 2003.

[15] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, no. 3, pp. 399–404, 1956.

[16] F. A. Kuipers, "An overview of algorithms for network survivability," *ISRN Communications and Networking*, vol. 2012, p. 19, 2012.

[17] G. Dantzig and D. R. Fulkerson, "On the max flow min cut theorem of networks," *Linear inequalities and related systems*, vol. 38, pp. 225–231, 2003.

[18] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[19] S. Yuan, S. Varma, and J. Jue, "Minimum-color path problems for reliability in mesh networks," *INFOCOM 2005*, pp. 2658–2669, 2005.
[20] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, no. 2, pp. 325–336, 1984.
[21] H. Nagamochi, K. Nishimura, and T. Ibaraki, "Computing all small cuts in an undirected network," *SIAM Journal on Discrete Mathematics*, vol. 10, no. 3, pp. 469–481, 1997.

APPENDIX A
SINGLE-LINK FAILURE

In this section, we assume that only 1 link in the network can fail at a time. Apart from that, we also assume that two successive failures occur with a time difference longer than the traversal time of any path in the network.

For this scenario, we present a polynomial-time algorithm to solve the ABPS problem. More specifically, in Algorithm 2, when $k = 1$, in Steps 1-2, we eliminate the links with availability less than δ, such that we obtain a new graph G' where each link has availability at least δ. Subsequently, by running Dijkstra's shortest path algorithm on G' from s to t, we can solve the ABPS problem for $k = 1$.

When $k \geq 2$, if the optimal solution consists of k fully link-disjoint paths, then 2 fully link-disjoint paths also exist and under the single-link failure scenario have availability is 1, which is optimal. Hence, by applying Suurballe's algorithm [20] in Step 3, the solution can be found, if it exists.

When the optimal solution consists of k partially link-disjoint paths, then 2 partially link-disjoint paths are also enough. The reason is that the availability of partially link-disjoint paths is decided by one unprotected link (say l). Hence, it suffices to find $k = 2$ partially link-disjoint paths. In Steps 4-5, for each link (u, v) whose availability is no less than δ, we create another (parallel) link between u and v with the same availability. After that, we call Suurballe's algorithm [20] from s to t. Since in the optimal solution the unprotected link has availability at least δ, by creating the parallel links whose availability is at least δ, the paths p_1 and p_2 returned by Suurballe's algorithm [20] are two "fully" link-disjoint paths. After that, if p_1 and p_2 traverse the parallel links, we then merge these two links into one link. This kind of link reflects the unprotected link in the optimal solution. On the other hand, the links whose availability is less than δ are protected in the returned solution because of the correctness of Suurballe's algorithm. Therefore, an optimal solution can be found by algorithm 2.

APPENDIX B
HARDNESS OF THE ABBPS PROBLEM FOR K=2

For some variants, like the fully link disjoint case, the ABBPS problem is polynomially solvable[4], however it is NP-hard in its general setting.

Theorem 10: The partially link-disjoint ABBPS problem for $k = 2$ is NP-hard.

Proof: We provide a proof for when $k = 2$. As it is shown in Fig. 16, assume we are given a path (denoted by GP) s-a-b-c-d-t with the availability labeled on each link and all the other links have an availability of 1. We now want to find a partially link-disjoint path with GP such that their combined availability is no less than 1. Since the requested availability is 1, only link (a, b) and link (c, d) can be unprotected in an optimal solution. Suppose that when link (a, b) is eliminated, there do not exist paths from node s to nodes b, c, d and t, and when link (c, d) is eliminated, there are no paths from node b to nodes d and t. In this context, to solve the partially link-disjoint ABBPS problem for $k = 2$, both (a, b) and (c, d) should be unprotected in the optimal solution. This is equivalent to finding three pairs of link-disjoint paths between node pairs (s, a), (b, c) and (d, t) (i.e., the Disjoint Connecting Paths problem). Hence, solving the partially link-disjoint ABBPS problem for $k = 2$ yields a solution to the NP-hard Disjoint Connecting Paths problem. ∎

[4]Trivially, by looking for an unprotected path with maximum availability in the network where all the links from the primary path are excluded.

Algorithm 2 ABPSSingleLinkFailure(G, s, t, δ, k)

1: Eliminate the links with availability less than δ on G, thereby obtaining a new graph G'.
2: Run Dijkstra's algorithm on G' from s to t. If the solution is found then return the result; Else if $k > 1$, continue; Otherwise output there is no solution.
3: Run Suurballe's algorithm [20] on G from s to t. Return the result if the solution is found, otherwise continue.
4: Create another (parallel) link between u and v with the same availability $A_{(u,v)}$, for each $(u, v) \in \mathcal{L}$ if $A_{(u,v)} \geq \delta$. The graph is denoted as G''.
5: Run Suurballe's algorithm [20] on G'' from s to t. Return the result if the solution is found, otherwise output there is no solution.

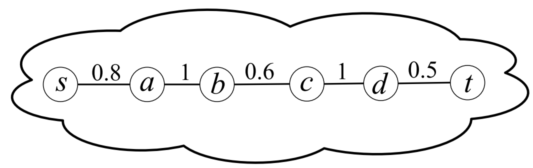


Fig. 16: Reduction of the ABBPS problem with partially link disjoint paths to the Disjoint Connecting Paths problem.