ExpectAll: A BDD Based Approach for Link Failure Resilience in Elastic Optical Networks

Gustav S. Bruhns, Martin P. Hansen, Rasmus Hebsgaard, Frederik M. W. Hyldgaard, and Jiří Srba

Department of Computer Science, Aalborg University, Aalborg, Denmark

Abstract. Constantly growing demands on higher bandwidth and quality of service in modern communication networks motivate the introduction of fully optical network technologies that can eliminate the bottlenecks of optical to digital signal conversions. Recent advances in elastic optical networks enable fine-grained resource allocation technologies for traffic demands, which introduces the Routing and Spectrum Allocation (RSA) problem. In order to improve network resilience for multiple link failures while avoiding double light-spectrum allocation, we present *ExpectAll*—a novel approach and a tool for resilience and path/spectrum allocation based on binary decision diagrams (BDDs). Our method efficiently computes and stores all solutions to the RSA problem in the BDD data structure, facilitating optimal and fast failover protection for failure scenarios even with multiple failing links. *ExpectAll* surpasses the state-of-the-art methods in both the speed of finding a single optimal solution for a currently occurring failure scenario as well as in the preparation time required to precompute all optimal route and spectrum assignments.

1 Introduction

With more than two-thirds of the global population having access to the internet and the increasing amount of data that originates from more and more devices being connected, modern data networks are put under pressure, heightening the need for increased bandwidth and network resilience [16]. Elastic, all-optical networks [10] are a possible solution to deal with these challenges. Traditionally, optical networks use wavelength-division multiplexing (WDM) [33] in order to split the frequency spectrum into slots of 50 GHz. As the amount of traffic has increased and is only predicted to increase more in the future, a new flexible paradigm has been proposed using elastic *Flexgrid* technology to enable more fine-grained splitting of the bandwidth down to 6.25 GHz slots [30]. In elastic optical networks, data is transported along *lightpaths*, which are all-optical connections between two access points in the network using one or more spectrum slots. Given a set of traffic demands, the routing and spectrum allocation (RSA) problem [31] involves finding a lightpath in the form of a route through the network and a set of spectrum slots for each demand. A solution to the RSA problem must comply with the constraints of:

- *Continuity*: A lightpath must use the same spectrum slots throughout its entire flow through the network.
- *Contiguity*: The spectrum slots used on a lightpath must be consecutive (follow each other).
- *Non-overlapping*: For each link in the network, a spectrum slot can be used by at most one lightpath.

With the increasing scale of networks and the amount of data that can be transported through an elastic optical network, the consequences of link outages become more severe [28,27]. Examples include loss of business revenue and disruption of safety-critical networks [12,17,24]. It is therefore important that networks quickly recover from link failures to reduce the consequences by finding an alternative routing for the demands affected by the link failures [19]. Current approaches in all-optical networks achieve quick recovery times for link failures by reserving a backup path for each demand [9,3] through over-allocation, thereby wasting network resources. Furthermore, this approach can only handle one-link failures, but multiple links are likely to fail [25]; hence, preparing for only one link failures can be inadequate. Ensuring resilience to multiple link failures with quick recovery times in elastic optical networks therefore presents a relevant challenge in the foreseeable future.

Our contributions We design and implement *ExpectAll*, a novel approach for solving the RSA problem using Binary Decision Diagrams (BDDs) to ensure failure resilience in elastic optical networks. The tool efficiently finds and compactly stores *all* solutions to the RSA problem, which can be leveraged to quickly provide real-time solutions to multiple link failure scenarios. To this end, our contributions are as follows.

First, we investigate the possible application of existing approaches relying on integer linear programming (ILP) for ensuring optimal failure resilience in optical networks. We experiment on two real network topologies in multiple-link failure scenarios and show that ILP is impractical for ensuring failure resilience, both in terms of the time for synthesizing new network configurations as well as in the excessive memory requirements.

As our second contribution, we leverage the BDD technology to solve the RSA problem for the purpose of failure resilience, culminating in an open-source tool *ExpectAll*. We prove that our approach finds all optimal solutions and present improvements to increase its scalability w.r.t. the number of traffic demands.

Finally, we showcase two applications of the BDD technology for ensuring resilience under multiple-link failures. The first application can handle an arbitrary number of link failures, whereas the second application provides quicker recovery times at the cost of being able to handle only an a priori given maximum number of failing links. The two applications are compared to the ILP approach, where it is clear that they both outperform the ILP solver when comparing how quickly they are able to recover from link failures, as well as how they scale when increasing the number of link failures.

Related work The Routing and Wavelength Assignment (RWA) problem is a well-studied [46,43,61] NP-complete problem [14] in optical networks that use Wavelength Division Multiplexing (WDM) technology [33] to partition the spectrum into a fixed number of wavelengths. Later on, the notion of fine-grained spectrum allocation was introduced for elastic optical networks as an improvement on the WDM optical networks, generalizing the problem to the routing and spectrum allocation (RSA). This more recent RSA problem is, as expected, also NP-hard [15].

Integer Linear Programming (ILP) has originally been used to formalize and solve the RSA lightpath constraints with the goal of minimizing the maximum slot index used in the light spectrum [15,58,42]. The ILP formulations were later improved to be more concise and efficient; example encodings are by Zhang et al. [62] improving the work done in [58], and Esteban et al. [56] who introduce even more efficient ILP encoding by employing the notion of channels to handle the spectrum contiguity constraint. Wang et al. [57] contribute with a relaxed ILP problem that establishes a lower bound of the optimal solution. In contrast to our approach, the ILP implementations are relatively effective in finding a single optimal solution to a given RSA problem. However, finding all optimal solutions (as needed for the failover protection) is not feasible using the ILP method as we demonstrate in our paper. While ILP formulations can find optimal solutions, they are generally not applicable for time-critical purposes, such as reacting to link failures in the order of miliseconds [19]. For more scalable approaches to solve the RSA problem, heuristics have been proposed [15,34,41], as well as genetic algorithms [30,29,39] and reinforcement learning models [54,60]. These approaches trade off optimality for computation speed, where the genetic algorithms tend to be closer to achieving more optimal solutions than the heuristics and reinforcement learning approaches at the cost of being computationally slower. In contrast to these approaches, we are able to find and efficiently store all optimal solutions for path and spectrum allocations, which can prove useful for handling changes during the network operation, such as quickly recovering from link failures.

While the study of fast failover protection [11] in IP networks [5,45], MPLS networks [44,55], Segment Routing networks [22,21] and software-defined networks [53,13] has been well-studied, these approaches cannot be easily transferred to elastic optical networks due to the orthogonally different way of packet forwarding in traditional networks (allowing us to e.g. analyze/modify packet headers) and fully optical ones. The current approaches for ensuring link failure resilience in optical networks generally only prepare for one-link failures by preallocating backup paths [9,3,23,50,51]. For example, Castro et al. [9] propose a MILP formulation that maximizes the total bitrate recovered in case of a single-link failure scenario by allocating a backup path for each demand such that when a demand is affected by a link failure, it can quickly switch over to its backup path, and Singhal et al. [51] and Gao et al. [23] expand upon this idea with the notion of a more resource-friendly cross-sharing, where groups of demands with link-disjoint primary paths are allowed to share a backup path.

These approaches based on preallocated backup paths assume only one link failure, but some research (see e.g. Athe and Singh [4] and Li et al. [40]) has been carried out to extend the failure resilience to two link and network-bound link failures. Common for the current approaches to ensuring failure resilience is that they require additional spectrum resources to always be allocated for both the primary and backup paths in the network to ensure quick failover, which entails that they do not provide optimal solutions. Our approach finds optimal solutions while being able to handle multiple link failures without having to resort to resource over-allocation.

Organization The rest of the paper is organized as follows. First, we formally define the RSA problem and the problem of handling link failures in Section 2. Then, in Section 3, we present and evaluate how to use an ILP formulation to handle link failures. In Section 4, we encode the RSA problem into BDDs which we then use in Section 5 to handle link failures. Finally, in Section 6, we compare our BDD-based approach with the ILP approach, and conclude on our work in Section 7.

2 Problem Definition

Let us first introduce formally the routing and spectrum allocation problem (RSA) and the problem of failover protection. We start with the definition of a network and a routing path.

A *network topology* is a tuple $G = (V, E, src, tgt)$ where V is a finite set of nodes, E is a finite set of edges and $src, tgt : E \to V$ denote the source and target of an edge, excluding self-loops by imposing $src(e) \neq tgt(e)$ for every $e \in E$. A *path* in G is a sequence of connected edges $\pi = e_1e_2e_3...e_n \in E^*$ such that $tgt(e_i) = src(e_{i+1})$ for all i, $1 \leq i < n$. Let $e \in \pi$ denote that an edge e is present in the path π and let $\pi \cap \pi'$ denote the set of all edges that the two paths share. Let $first(\pi) = e_1$ and $last(\pi) = e_n$ be mappings that return the first and the last edge on the path π, respectively. A *simple path* is a path where $src(e') \neq src(e)$ and $tgt(e) \neq tgt(e')$ for all pairs of distinct edges $e, e' \in \pi$; let $\boldsymbol{Paths}$ be the set of all simple paths in the topology G.

We assume a finite set of *demands* D with source and target nodes represented by the mappings $ingress, egress : D \to V$ respectively, and $size : D \to \mathbb{N}$ representing the amount of data of a given demand. Furthermore, we assume a finite set of *spectrum slots* $F = \{1, 2, .., f_{max}\}$. Finally, a *channel* is any subset $C \subseteq F$ of consecutive slots, and let $\mathbb{C}$ be the set of all channels. All possible channels that can be used to transfer a given demand are represented by the mapping $channels : D \to 2^{\mathbb{C}}$.

Definition 1 (Routing and Spectrum Allocation Problem). *Given an input consisting of*

- *a network topology* $G = (V, E, src, tgt)$ *with simple paths* $\boldsymbol{Paths}$,
- *a set of demands* $D = \{d_1, d_2, ..., d_m\}$,

- *a mapping* $DPaths : D \to 2^{\boldsymbol{Paths}}$ *of available paths for each demand* $d \in D$*, where for every* $\pi \in DPaths(d)$ *it holds that* $src(first(\pi)) = ingress(d)$ *and* $tgt(last(\pi)) = egress(d)$*,*
- *a finite set of available slots* $F = \{1, 2, .., f_{max}\}$*, and*
- *a modulation mapping* $\Delta : \boldsymbol{Paths} \to \mathbb{N}$ *which returns the number of slots required per sent unit of data on a given path,*

our task is to find a solution (P, ω) *where*

- $P : D \to \boldsymbol{Paths}$ *is the route assignment function such that* $P(d) \in DPaths(d)$ *for every demand* $d \in D$*, and*
- $\omega : D \to 2^F$ *is the spectrum allocation function such that* $\omega(d) \in channels(d)$ *and* $|\omega(d)| = \Delta(P(d)) \cdot size(d)$ *meaning that the allocated channel has enough spectrum slots depending on the chosen modulation and the size of the demand,*

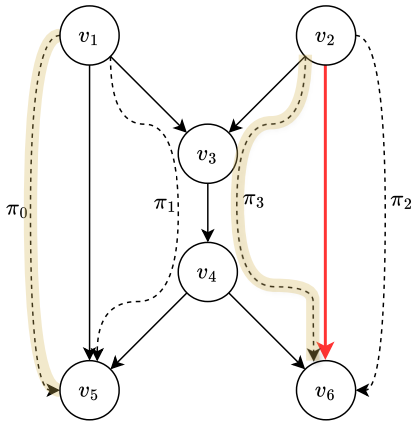
such that for all pairs of distinct demands $d, d' \in D$ *either* $P(d) \cap P(d') = \emptyset$ *or* $\omega(d) \cap \omega(d') = \emptyset$*, i.e. either the chosen paths do not intersect or, if they do, then the allocated frequency slots may not overlap.*

Let us remark that the input to the RSA problem contains also the set of allowed paths for each demand (typically the shortest paths from the ingress to egress nodes). This allows us to restrict the routing to an (often small) subset of simple paths and we assume that these can be enumerated to become a part of the input.

Different metrics have been used to define optimal solutions to the RSA problem, such as minimizing unserved bandwidth of demands [56], the number of frequency slots used [15], and the highest frequency slot index used by any demand [42]. We are aiming to minimize the highest used frequency slot as this supports our intended application discusssed later on. Hence for a given spectrum allocation function ω, let $usage(\omega) = \max\{f \in \omega(d) \mid d \in D\}$ define the highest used frequency slot in ω. A *network optimal solution* is then a solution (P, ω) to the RSA problem where $usage(\omega) \leq usage(\omega')$ for any other solution to the RSA problem (P', ω').

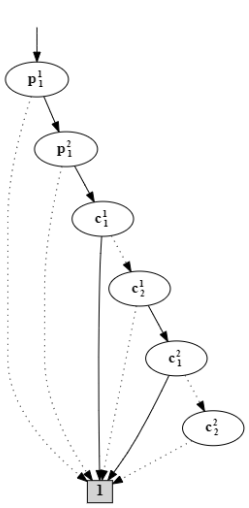
Example 1. Figure 1a illustrates a simple example of the RSA problem with a small network topology consisting of six nodes and seven edges and a spectrum width of two frequency slots. There are two demands, d_1 from v_1 to v_5 and d_2 from v_2 to v_6, both of size 1. Each demand has two possible paths. If the demand d_1 uses the longer path π_1, then it must be allocated two frequency slots due to the modulation. The same holds true for the demand d_2 if it uses the path π_3. Hence, there are three different possible channels for both demands, as seen in Figure 1a. An example of an optimal solution to this RSA problem is Solution 1 where the demand d_1 is assigned the path π_0 and channel $\omega(d_1) = \{1\}$ while the demand d_2 is assigned the path π_2 and channel $\omega(d_2) = \{1\}$. Solution 2 is not optimal as its highest used frequency slot is 2.

Having introduced the RSA problem, we now formally define the problem of protecting a network for up to k link failures.

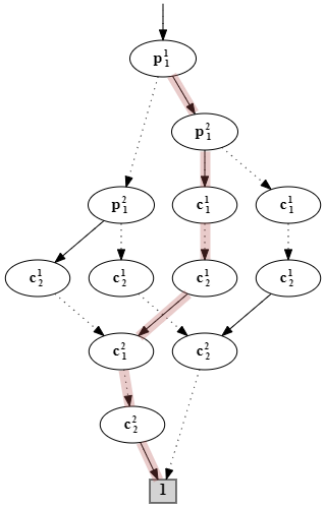


Problem:	Solution 1 (optimal):
$D = \{d_1, d_2\}$	$P(d_1) = \pi_0$
$size(d_1) = size(d_2) = 1$	$P(d_2) = \pi_2$
$DPaths(d_1) = \{\pi_0, \pi_1\}$	$\omega(d_1) = \{1\}$
$DPaths(d_2) = \{\pi_2, \pi_3\}$	$\omega(d_2) = \{1\}$
$\Delta(\pi) = \begin{cases} 2 & \text{if } \pi \in \{\pi_1, \pi_3\} \\ 1 & otherwise \end{cases}$	Solution 2:
$F = \{1, 2\}$	$P(d_1) = \pi_0$
$\mathbb{C} = \{\{1\}, \{2\}, \{1, 2\}\}$	$P(d_2) = \pi_3$
$channels(d_1) = \{\{1\}, \{2\}, \{1, 2\}\}$	$\omega(d_1) = \{1\}$
$channels(d_2) = \{\{1\}, \{2\}, \{1, 2\}\}$	$\omega(d_2) = \{1, 2\}$

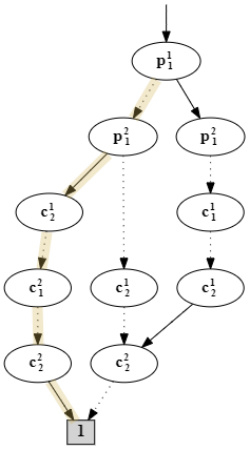
(a) Example of RSA problem and two corresponding solutions. Solution 2 is highlighted in yellow, and uses 2 slots. Solution 1 uses only 1 slot. In the case where the thick red edge fails, only solution 2 is valid.



(b) *noClash*



(c) *assignment*



(d) *rsa-all* = *assignment* ∧ *noClash*

Fig. 1: BDD for *noClash*, *assignment*, and *rsa-all* for the example shown in Figure 1a. The red highlighted path in 1c shows the path assignments $P(d_1) = \pi_1$ and $P(d_2) = \pi_3$, and the channel assignments $\omega(d_1) = \omega(d_2) = \{1, 2\}$ in which the two demands clash. Clashing assignments are filtered in the BDD *rsa-all*. The yellow highlighted path in Figure 1d corresponds to two solutions: Solution 2 from Figure 1a and a similar one where we instead have $\omega(d_1) = \{2\}$.

Definition 2 (k-Link Failover Problem). *Given a set of link failures $E_{fail} \subseteq E$ where $|E_{fail}| \leq k$, we want to find a solution (P, ω) to the RSA problem where $P(d) \cap E_{fail} = \emptyset$ for all $d \in D$.*

Example 2. Let us assume that the link between v_2 and v_6 has failed in our running example from Figure 1a. Our goal is to find a solution where no demand is assigned a path that uses the failed link. Clearly, Solution 1 from Figure 1a is now not a valid option anymore. However, Solution 2 is still a valid solution as the demand d_2 is assigned the path π_3 which does not use the failed link. Now Solution 2 becomes an optimal solution in this failure scenario.

The goal is now to be able to quickly react to all possible k-link failure scenarios and suggest an alternative solution that does not use any of the failed links. In the next section, we shall discuss the possible applicability of integer linear programming to adress the failover problem.

3 Failure Resilience via ILP

The goal of fast failover resiliece is to provide an optimal RSA solution for any failure scenario. According to the Metro Ethernet Forum [19], we aim for an average reaction time of less than 50 ms with an upper limit of 200 ms when link failures occur. In order to achieve this goal, we first examine whether the classical approach via integer linear programming (ILP) can be employed. We implement a slight modification (relaxation of the requirement of biderectional demands) of the state-of-the-art ILP formulation of the RSA problem from [42]. The ILP program presented in [42] minimizes for the highest used spectrum slot. For completeness, we now present the implemented encoding that uses the integer variables $x_{d\pi f} \in \{0, 1\}$ where $x_{d\pi f} = 1$ signifies that the demand d uses the path $\pi \in DPaths(d)$ with the start frequency slot $f \in F$. Additionally, it uses the parameters $n_{d\pi}$ to denote the number of frequency slots that the demand d needs to be transmitted along the path $\pi \in DPaths(d)$, i.e. $n_{d\pi} = size(d) \cdot \Delta(\pi)$. The ILP formulation is then as follows:

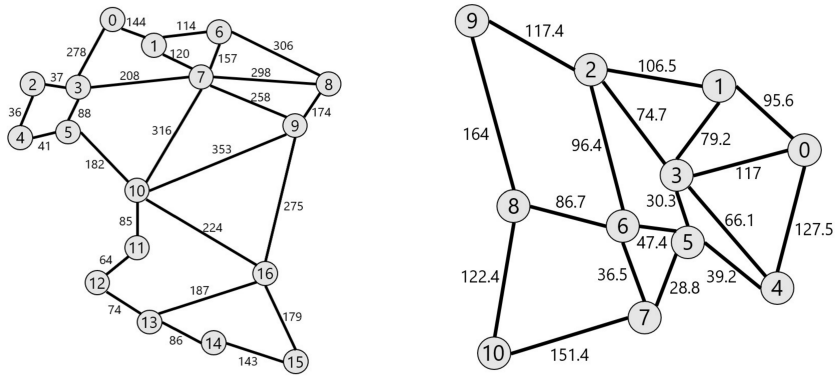
$$minimize\ f_{max} \tag{1}$$

$$\sum_{f \in F} \sum_{\pi \in DPaths(d)} x_{d\pi f} = 1, \qquad \forall d \in D \tag{2}$$

$$\sum_{d \in D} \sum_{\substack{\pi \in DPaths(d) \\ e \in \pi}} \sum_{f - n_{d\pi} + 1 \leq f' \leq f} x_{d\pi f'} \leq 1, \qquad \forall e \in E, \forall f \in F \tag{3}$$

$$\sum_{f \in F} (f + n_{d\pi} - 1) \cdot x_{d\pi f} \leq f_{max} \qquad \forall d \in D, \forall \pi \in DPaths(d) \tag{4}$$

Constraint 2 assigns a single path and a start slot to each demand. Constraint 3 ensures that there is no spectrum clash between demands on any edge



(a) Deutsche Telecom Backbone (DT) [36] (b) Kanto 11 Network [36]

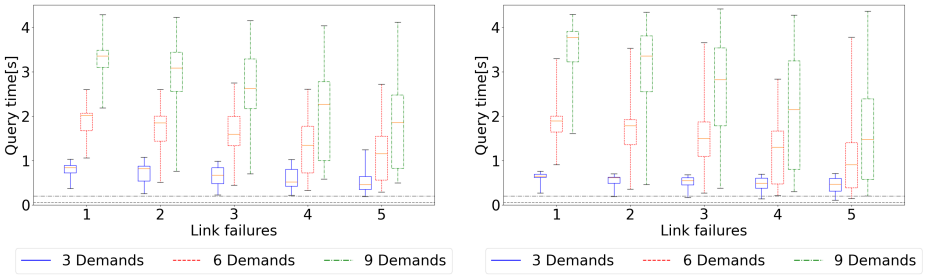
Fig. 2: Evaluation networks; numbers on edges indicate the distance in kilometers

of the paths. Finally, Constraint 4 checks that the slots used by the demands are not exceeding the variable f_{max} (highest used frequency slot) that we are minimizing.

When link failures occur in the network, the ILP formulation can be used to compute an optimal route and spectrum assignment that avoids the failed links. We evaluate this approach through an experiment, using the Gurobi ILP solver in Python [26], and run the experiment on a Ubuntu 18.04.5 cluster with 2.3 GHz AMD Opteron 6376 processors, with a memory limit of 30GB. For the experiment, we use the Deutsche Telecom Backbone (DT) and Kanto 11 network topologies (see Figure 2), two highly connected topologies where nodes/cities have their correct population sizes as referenced in [59,49]. The population sizes are used to generate demands using the standard gravity model [48]. Two shortest paths are generated for each demand using the semi-disjoint path generation algorithm from [32], and we assume a standard 320 slot spectrum capacity [36] with modulation 1 for all paths.

We generate random k-link failure scenarios with uniform probability of link failures and Figures 3a and 3b show box plots of the time it takes to compute a new solution for 1-5 link failures for 3, 6, and 9 demands in the DT network and Kanto network respectively. We see that the response time for finding a solution to the problem is in seconds, even for a smaller number of demands, clearly exceeding the expected average of 50 ms response time. As a conclusion, the ILP approach is too slow (by several orders of magnitude) in order to be applied for fast failover recovery.

As an alternative approach, one can consider to precompute and store all possible optimal solutions for any k-or-fewer link failure scenarios such that a solution can be provided almost instantaneously once we encounter link failures. This implies that for each possible combination of failed edges $E_{fail} \subseteq E$ where $|E_{fail}| \leq k$, we need to precompute a solution using the ILP formulation. This



(a) DT network (b) Kanto network

Fig. 3: Box plots of ILP query times based on 1000 random link-failure scenarios; dotted and dashed vertical lines show the 50 ms and 200 ms threasholds

Table 1: Estimated precomputation time for failure resilience using ILP

DT/Kanto		Demands		
		3	6	9
Failures	1	42s/23s	2m/2m	3m/3m
	2	17m/7m	40m/19m	2h/35m
	3	5h/2h	11h/4h	17h/6h
	4	2d/10h	5d/23h	8d/2d
	5	19d/3d	39d/5d	60d/8d

approach clearly does not scale (neither in CPU time nor the required memory) due to its high combinatorial complexity. Table 1 shows the estimated precomputation times, based on the average of 1000 randomly generated failure scenarios, for up to 9 demands and 5 link failures (the first number is for DT and the second for Kanto). We can observe that already for three failing links, the precomputation takes hours, for four link failures days and for five several weeks, even for as little as three demands. Additionally, the memory required to store all these solutions grows exponentially too. We hence conclude that precomputing solutions to link failures using ILP is not feasible either.

In the next sections, we present a different method based on binary decision diagrams to efficiently compute and store all solutions to a given RSA problem; our method scales orders of magnitude better than the ILP approach, both in the reaction time as well as the precomputation time/memory.

4 BDD Encoding of RSA

We shall now discuss our method and tool *ExpectAll* that relies on binary decision diagrams for computing and compactly storing all solutions to the RSA problem. Binary Decision Diagrams (BDDs) were introduced in [1] and [38] as a data

structure to efficiently represent and manipulate Boolean functions. The concept was later refined in [7] with more efficient Boolean operations. As such, BDDs have been revolutionary at efficiently representing finite state spaces [8] and have by now been applied in various problem domains [28,27,37,18,47].

A BDD is a rooted, directed and acyclic graph [2]. Every non-leaf node in a BDD is labeled with a Boolean variable, while there are exactly two leaf nodes labelled with the value 0 (*False*) and 1 (*True*). Each non-leaf node u has two outgoing edges denoted as $low(u)$ and $high(u)$; following an edge corresponds to setting the variable at node u to either *False* (low edge) or *True* (high edge). The edges are commonly visualized by using a solid line for $high(u)$ and a dotted line for $low(u)$. In the graphical notation, the leaf node 0 and all its incomming arcs are usually omitted. An example of a BDD can be seen in Figure 1b. Following the right-most branch from the root to the terminal node 1 represents a single assignment where all variables are true except for $\mathbf{c_1^1}$, $\mathbf{c_1^2}$ and $\mathbf{c_2^2}$. The left-most branch, on the other hand, represents a whole set of assignments where $\mathbf{p_1^1}$ is false and all other variables can be set arbitrarily.

A BDD is ordered (OBDD) if the variables in the BDD come in the same order $x_1 < x_2 < ... < x_n$ on all paths from the root to a leaf. An OBDD can be reduced by merging nodes with identical subgraphs, and by deleting nodes where the subgraphs for $low(u)$ and $high(u)$ are equivalent. Such a BDD is called a reduced OBDD (ROBDD) [2]. In the rest of the paper, we write simply BDD instead of ROBDD. In our encoding, we shall use a BDD extension that supports first-order quantifiers and is closed under all Boolean operations and quantifiers; for details see e.g. [38].

4.1 Representing a set as a Boolean function

Given a finite set $S = \{s_0, s_1, ..., s_{|S|-1}\}$, let $\overline{\mathbf{x}} = [\mathbf{x_k}, ..., \mathbf{x_1}]$ be a vector of Boolean variables where $k = \lceil log_2(|S|) \rceil$. Now any truth assignment α to $\overline{\mathbf{x}}$ can be interpreted as a natural number $n(\alpha) \in \mathbb{N}$ written in binary notation. Thus $\overline{\mathbf{x}}$ encodes the $n(\alpha)$'th element of S. Let $\overline{\mathbf{x}}(s)$ denote the Boolean expression over $\overline{\mathbf{x}}$ with just the single truth assignment corresponding to the element s. Given a Boolean expression $b(\overline{\mathbf{x}})$, let $[\![b(\overline{\mathbf{x}})]\!]$ denote the encoded subset $\{s_{n(\alpha)} \mid \alpha \text{ satisfies } b(\overline{\mathbf{x}})\} \subseteq S$ such that $[\![b(\overline{\mathbf{x}})]\!]$ is a set consisting of all elements that are encoded by all possible Boolean assignments satisfying the expression b.

Example 3. Let us consider the set of paths $\{\pi_0, \pi_1, \pi_2, \pi_3\}$ from Figure 1a. We need two boolean variables $\overline{\mathbf{p}} = [\mathbf{p_2}, \mathbf{p_1}]$ to encode any of the given paths. For instance, we encode the path π_0 by $\overline{\mathbf{p}}(\pi_0) = \neg\mathbf{p_2} \wedge \neg\mathbf{p_1}$. In a Boolean expression such as $b = \mathbf{p_1}$, where the variable $\mathbf{p}_2$ is free, the Boolean function $b(\overline{\mathbf{p}})$ is satisfied both when $[\mathbf{p_2} \mapsto 0, \mathbf{p_1} \mapsto 1]$ and $[\mathbf{p_2} \mapsto 1, \mathbf{p_1} \mapsto 1]$, which means $[\![b(\overline{\mathbf{p}})]\!] = \{\pi_1, \pi_3\}$.

4.2 Construction of BDDs with all valid RSA assignments

We shall now describe how, for a given instance of RSA problem, we construct a BDD representation of all its solutions. As shown in Table 2, for each demand d,

Table 2: Variables used in the BDD encodings of RSA problem

Variables	Description
$\overline{\mathbf{c}^d} = [\mathbf{c}^d_n, \mathbf{c}^d_{n-1}, ..., \mathbf{c}^d_1]$ where $n = \lceil \log_2(\lvert channels(d) \rvert) \rceil$	Encoding of $channels(d)$
$\overline{\mathbf{p}^d} = [\mathbf{p}^d_n, \mathbf{p}^d_{n-1}, ..., \mathbf{p}^d_1]$ where $n = \lceil \log_2(\lvert DPaths(d) \rvert) \rceil$	Encoding of paths $DPaths(d)$

we use a vector of Boolean variables $\overline{\mathbf{p}^d}$ to encode the path assigned to demand d, and another vector of Boolean variables $\overline{\mathbf{c}^d}$ to encode the channel assigned to demand d. To encode a solution (P, ω) to the RSA problem (see Definition 1), we use the vector of vectors $\overline{\mathbf{p}^D} = [\overline{\mathbf{p}^{d_1}}, \overline{\mathbf{p}^{d_2}}, ..., \overline{\mathbf{p}^{d_{|D|}}}]$ to encode P and the vector of vectors $\overline{\mathbf{c}^D} = [\overline{\mathbf{c}^{d_1}}, \overline{\mathbf{c}^{d_2}}, ..., \overline{\mathbf{c}^{d_{|D|}}}]$ to encode ω.

Example 4. We can see that the BDD `assignment` shown in Figure 1c represents all $3^2 = 9$ possible combinations of P and ω for the running example. For example the assignment where $P(d_1) = \pi_1$, $P(d_2) = \pi_3$ and $\omega(d_1) = \omega(d_2) = \{1, 2\}$ is highlighted in red.

We now define three BDDs which enforce that $\overline{\mathbf{p}^D}$ and $\overline{\mathbf{c}^D}$ encode only valid solutions of the RSA problem. Specifically, the BDDs must ensure that each demand d is assigned a path $\pi \in DPaths(d)$ and a channel $C \in channels(d)$ such that $|C| = \Delta(\pi) \cdot size(d)$. Moreover, whenever a demand shares an edge with another demand, they cannot be assigned overlapping channels.

To enforce that no two demands are allowed to clash, we define the BDD `noClash` (Figure 1b) by

$$\texttt{noClash}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}) = \tag{5}$$
$$\bigwedge_{\substack{d,d' \in D,\\ d \neq d'}} \bigwedge_{\substack{\pi \in DPath(d),\\ \pi' \in DPaths(d'),\\ \pi \cap \pi' \neq \emptyset}} \left(\neg\big(\overline{\mathbf{p}^d}(\pi) \wedge \overline{\mathbf{p}^{d'}}(\pi')\big) \vee \bigwedge_{\substack{C \in channels(d),\\ C' \in channels(d'),\\ |C| = \Delta(\pi) \cdot size(d),\\ |C'| = \Delta(\pi') \cdot size(d'),\\ C \cap C' \neq \emptyset}} \neg\big(\overline{\mathbf{c}^d}(C) \wedge \overline{\mathbf{c}^{d'}}(C')\big) \right)$$

satisfying that $(P, \omega) \in [\![\texttt{noClash}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})]\!]$ iff for all $d, d' \in D$ where $d \neq d'$ either $P(d) \cap P(d') = \emptyset$ or $\omega(d) \cap \omega(d') = \emptyset$.

Example 5. Consider again the network from Figure 1a. There is only one way the two demands can clash. The BDD `noClash` therefore has to encode that it is satisfied by all path and channel assignments, except when d_1 and d_2 are assigned the paths π_1 and π_3 respectively, and the channels $\omega(d_1) = \omega(d_2) = \{1, 2\}$. As shown in Figure 1b, the BDD `noClash` encodes exactly this, as the only way not to satisfy this BDD is taking the right-most path down to the node labeled $\mathbf{c}^2_2$, and then set the value of the Boolean variable $\mathbf{c}^2_2$ to *True*, corresponding to this single clashing assignment.

Next, we must enforce that each demand d is assigned a correct path and channel assignment pair. To this end, we define the BDD $\mathit{assignment}$ as

$$\mathit{assignment}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}) = \bigwedge_{d \in D} \bigvee_{\pi \in DPath(d)} \left(\overline{\mathbf{p}^d}(\pi) \wedge \Big(\bigvee_{\substack{C \in channels(d), \\ |C| = \Delta(\pi) \cdot size(d)}} \overline{\mathbf{c}^d}(C) \Big) \right) \quad (6)$$

and clearly $(P, \omega) \in [\![\mathit{assignment}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})]\!]$ iff for all $d \in D$, it holds that $P(d) \in DPaths(d)$, $\omega(d) \in channels(d)$ and $|\omega(d)| = \Delta(P(d)) \cdot size(d)$.

Finally, we use $\mathit{assignment}$ and $\mathit{noClash}$ to define the BDD $\mathit{rsa\text{-}all}$ as

$$\mathit{rsa\text{-}all}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}) = \mathit{assignment}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}) \wedge \mathit{noClash}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}). \quad (7)$$

Example 6. In our running example, we can see in Figure 1d that $\mathit{rsa\text{-}all}$ contains all valid solutions for the given RSA problem. All the assignments from the BDD $\mathit{assignments}$ are valid solutions, except the assignment that is marked red in Figure 1c, since it is the only assignment that does not satisfy the BDD $\mathit{no\text{-}clash}$, as noted in Example 5. Hence, this is the only assignment that does not appear in $\mathit{rsa\text{-}all}$. The highlighted path in $\mathit{rsa\text{-}all}$ corresponds to the two valid solutions, one solution being Solution 2 from Figure 1a, the other being the same as Solution 2 except for that the demand d_1 is assigned the channel $\{2\}$ instead of $\{1\}$.

We can now state the correctness theorem of our BDD construction.

Theorem 1. *The pair (P, ω) is a solution to the RSA problem iff $(P, \omega) \in [\![\mathit{rsa\text{-}all}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})]\!]$.*

Proof (Proof sketch). " $\Longrightarrow$ " Assume (P, ω) is a solution to the RSA problem. By Definition 1, the encodings $\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}$ of (P, ω) satisfy both the constraints enforced by $\mathit{assignment}$ and $\mathit{noClash}$, and it thus follows that $(P, \omega) \in [\![\mathit{rsa\text{-}all}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})]\!]$.

" $\Longleftarrow$ " Let $(P, \omega) \in [\![\mathit{rsa\text{-}all}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})]\!]$. By the constraints enforced by $\mathit{assignment}$ we know that each demand has been assigned a valid path and channel combination given the used modulation, and due to the constraints enforced by $\mathit{noClash}$ we know that the path and channel assignments given to each demand result in a solution, with no clashing between any of the demands. Hence (P, ω) is a solution to the RSA problem. □

As the BDD $\mathit{rsa\text{-}all}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})$ now contains all solutions to the RSA problem, it is possible for a network operator to query it to find for example an optimal solution or a solution matching any particular property desired by the operator. For the k-link failover problem, there is no need to store all possible solutions, as we need only to represent all optimal path and channel assignments for each possible scenario of up to k link failures. To limit the set of solutions,

we can disallow a fragmenation of the spectrum space i.e. restrict possible gaps of slots between the assigned channels. Concretely, we require that a channel assignment ω must assign all slots smaller than or equal to $usage(\omega)$ to at least one demand. This is enforced by specifying that the channel assigned to a demand either uses the first slot, or follows directly after a channel assigned to another potentially path-overlapping demand.

Definition 3 (Gap-free solution). *A solution to the RSA problem (P, ω) is* gap-free *if for all $d \in D$, either $min(\omega(d)) = 1$ or there exists a $d' \in D, \pi' \in DPaths(d')$ and $\pi \in DPaths(d)$ s.t. $\pi \cap \pi' \neq \emptyset$ and $min(\omega(d)) = max(\omega(d'))+1$.*

Theorem 2. *If (P, ω) is a solution to an RSA problem, then there exists a gap-free solution (P, ω') such that $usage(\omega') \leq usage(\omega)$.*

Proof. Assume a channel allocation ω for the demands $d_1, d_2, ..., d_m$ such that $d_i \leq d_j$ iff $min(\omega(d_i)) \leq min(\omega(d_j))$. A new ω' can now be constructed, such that ω' is *gap-free*. For d_i, let D_i be the set of demands $d_k < d_i$ where $\omega(d_k) \cap \omega(d_i) = \emptyset$ and there exists $\pi_k \in DPaths(d_k)$ and $\pi_i \in DPaths(d_i)$ s.t. $\pi_k \cap \pi_i \neq \emptyset$. Let ω' initially be undefined for all demands. We now add each demand to ω' one by one in the specified order, such that if $D_i = \emptyset$ then $\omega'(d_i) = \{1, ..., |\omega(d_i)|\}$, otherwise let the highest slot assigned by ω' to a demand in D_i be $\Omega = \max_{d \in D_i}(max(\omega'(d)))$ and then the new channel of d_i is specified as $\omega'(d_i) = \{\Omega + 1, \Omega + 2, ..., \Omega + |\omega(d_i)|\}$. It follows that $usage(\omega') \leq usage(\omega)$ since at any step $min(\omega'(d_i)) \leq 1 + |\bigcup_{d \in D_i} \omega(d)|$. □

The gap-free property can be imposed on the solutions encoded by the BDD `rsa-all` using the following `gapfree` BDD defined as

$$\texttt{gapfree}(\overline{\mathbf{c}^D}) = \tag{8}$$
$$\bigwedge_{d \in D} \left(\Big(\bigvee_{\substack{C \in channels(d), \\ min(C)=1}} \overline{\mathbf{c}^d}(C) \Big) \vee \Big(\bigvee_{\substack{d' \in D, \\ \exists \pi \in DPaths(d), \\ \exists \pi' \in DPaths(d'), \\ \pi \cap \pi' \neq \emptyset}} \; \bigvee_{\substack{C \in channels(d), \\ C' \in channels(d'), \\ min(C)=max(C')+1}} \overline{\mathbf{c}^d}(C) \wedge \overline{\mathbf{c}^{d'}}(C') \Big) \right)$$

where we have $\omega \in [\![\texttt{gapfree}(\overline{\mathbf{c}^D})]\!]$ iff ω satisfies the gap-free property as described in Definition 3. Additionally, since the gap-free property preserves an optimal solution for each routing assignment, it can handle the same link failure scenarios as `rsa-all`.

We now define a new BDD `rsa-gapfree` using `gapfree` as follows

$$\texttt{rsa-gapfree}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}) = \tag{9}$$
$$\texttt{assignment}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}) \wedge \texttt{gapfree}(\overline{\mathbf{c}^D}) \wedge \texttt{noClash}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})$$

and clearly $(P, \omega) \in [\![\texttt{rsa-gapfree}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})]\!]$ iff (P, ω) is a valid solution to the RSA problem and ω satisfies the gap-free property. Since `assignment`, `gapfree`

and *noClash* are just Boolean expressions, their order in the conjunction is semantically irrelevant; in the implementation, the conjunction is evaluated in a left-to-right order, resulting in an improved performance.

In addition to removing spectrum gaps, any optimal solution will never occupy any spectrum slot higher than $max_f = \sum_{d \in D} \max_{\pi \in DPaths(d)} \big(\Delta(\pi) \cdot size(d)\big)$ and thus all channels where $max(C) > max_f$ can be disregarded. By enforcing this upper limit, we effectively prune the candidate channels for each demand (and hence improve the performance) without losing optimality. A more greedy approach to approximate the candidate channels is to assign channels based on an established ordering of the demands. The idea is to remove symmetry in the channel assignments by imposing the following property.

Definition 4 (Limited). *Let $D = \{d_1, d_2, ..., d_m\}$ be a list of demands in a fixed ordering. A solution to the RSA problem (P, ω) is* limited *if $min(\omega(d_i)) \leq c_{max} + \sum_{j<i} |\omega(d_j)|$ for all $d_i \in D$, where $c_{max} = \max_{d \in D} |\omega(d)|$.*

This means that the demand d_i should start at a frequency slots that is no larger than the sum of cardinalities of the already scheduled frequency slots plus the size of the largest possible demand. Both the maximum spectrum slot limit as well as the limited property can be applied by restricting the mapping *channels* in the definition of *gapfree* in Equation 8. We note that the greedy approach can in some rare instances make it impossible to find a channel assignment for a particular path assignment, however, we never experienced this issue on any of the topologies from the Topology Zoo benchmark [35] (with more than 250 ISP topologies) as long as the demands are ordered from the largest one to the smallest. From now on, we shall use $rsa(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})$ to refer to the BDD from Equation 9 where both the limited property as well as the highest spectrum slot bound are applied.

4.3 Extraction of optimal solutions from a BDD

Once we built a BDD representing valid solutions to an RSA problem, we are interested in finding an optimal solution that minimizes the highest used spectrum slot. To this end, we introduce a new vector of Boolean variables $\overline{\mathbf{s}^F} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{f_{max}}]$ with the meaning that $\overline{\mathbf{s}^F}$ encodes a subset $F' \subseteq F$ of spectrum slots such that $f \in F'$ iff $\mathbf{s}_f$ is true. The intuition behind the use of these variables is that if a variable $\mathbf{s}_f$ is false, then we know that no demand has been assigned a channel with a slot greater than or equal to f, meaning that the usage is at most $f - 1$. We can enforce this using the BDD *rsa-slotBound* defined as

$$rsa\text{-}slotBound(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}, \overline{\mathbf{s}^F}) \tag{10}$$
$$= rsa(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}) \wedge \left(\bigwedge_{d \in D} \bigvee_{C \in channels(d)} \left(\overline{\mathbf{c}^d}(C) \wedge \bigwedge_{f \leq max(C)} \mathbf{s}_f \right) \right)$$

and we observe that $(P, \omega, F') \in [\![\texttt{rsa-slotBound}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}, \overline{\mathbf{s}^F})]\!]$ iff $(P, \omega) \in [\![\texttt{rsa}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})]\!]$ and F' contains all $f \leq usage(\omega)$.

An optimal solution can now be found by iteratively (e.g. using the binary search) identifying the smallest value of $f \in F$ for which we can still find a solution

$$(P, \omega, F') \in [\![\texttt{rsa-slotBound}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}, \overline{\mathbf{s}^F})]\!]$$

where the corresponding variable $\mathbf{s}_f$ can take the value false, i.e. $f \notin F'$, since it can then be inferred that the $usage(\omega) = f - 1$.

5 Failure Resilience via BDD Encoding

As shown in the previous section, we can build a BDD that stores all optimal solutions of the RSA problem. We now describe how to quickly extract optimal solutions from such a BDD for the purpose of providing time-critical responses to multiple links failing. In particular, the BDD $\texttt{rsa}$ as defined in Section 4.2 contains at least one solution for all failure scenarios, if such a solution exists. Furthermore, we know that at least one of these solutions is optimal under the corresponding failure scenario. To extract these solutions, we present two approaches. The first method supports an arbitrary number of link failures based on the idea of deleting solutions that use invalid paths given the link failures. The second method uses a precomputation in which link failures are directly encoded into the BDD.

5.1 Pruning by deletion

Solutions encoded in a given BDD $\texttt{rsa}$ that use invalid paths based on the set of link failures E_{fail} are no longer valid solutions. The invalid solutions can be deleted from $\texttt{rsa}$ as follows

$$\texttt{path-pruned-rsa}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}) = \texttt{rsa}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}) \wedge \bigwedge_{d \in D} \bigwedge_{\substack{\pi \in DPaths(d), \\ \exists e \in E_{fail}, \\ e \in \pi}} \neg \overline{\mathbf{p}^d}(\pi) \quad (11)$$

and clearly $(P, \omega) \in [\![\texttt{path-pruned-rsa}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})]\!]$ iff $(P, \omega) \in [\![\texttt{rsa}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})]\!]$ and it holds for all $d \in D$ that $P(d) \cap E_{fail} = \emptyset$. Since there is no limitation on the size E_{fail}, this pruning method can be used to prune the BDD for an arbitrary number of failed links.

5.2 Pruning by precomputation

As failure scenarios with a high number of concurrently failing links are less likely, we can focus on preparing a failover protection for an a priori given maximum number k of failing links. We hence construct a parameterized BDD with

additional variables representing the failed edges; these can be specified to retrieve the valid solutions for any given link failure scenario up to some sufficiently large k.

Let $E_u = E \cup \{e_{unused}\}$ where the auxiliary edge e_{unused} signifies a non-failing link. For a given k-link failover problem, we introduce a vector of k variable encodings $\overline{\mathbf{e}^K} = [\overline{\mathbf{e}^1}, ..., \overline{\mathbf{e}^k}]$ such that $\overline{\mathbf{e}^i} = [\mathbf{e}^i_n, \mathbf{e}^i_{n-1}, ..., \mathbf{e}^i_1]$ for $i \in K = \{1, 2, ..., k\}$, where $n = \lceil \log_2(|E_u|) \rceil$. The variable $\overline{\mathbf{e}^i}$ thus either encodes a specific link-failed edge or the auxiliary edge e_{unused} signifying that it does not encode any link failing. This makes it possible to encode up to k link failures, rather than being limited to exactly k links failing.

We shall first define a BDD $\mathtt{path\text{-}edge\text{-}overlap}$ to associate each path with its edges

$$\mathtt{path\text{-}edge\text{-}overlap}(\overline{\mathbf{p}^D}, \overline{\mathbf{e}}) = \bigvee_{d \in D} \bigvee_{\pi \in DPaths(d)} \bigvee_{e \in \pi} \overline{\mathbf{p}^d}(\pi) \wedge \overline{\mathbf{e}}(e) \tag{12}$$

where $(P, e) \in [\![\mathtt{path\text{-}edge\text{-}overlap}(\overline{\mathbf{p}^D}, \overline{\mathbf{e}})]\!]$ iff there exists a $\pi \in P(D)$ such that $e \in \pi$.

We can now define the BDD $\mathtt{failover}^k$ that encodes all valid path assignments for every combination of k or fewer link failures:

$$\begin{aligned}\mathtt{failover}^k(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}, \overline{\mathbf{e}^K}) = \mathtt{rsa}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}) \ \wedge \bigwedge_{m<j\leq k} \overline{\mathbf{e}^j}(e_{unused}) \\ \bigvee_{\substack{E'=\{e_1,e_2,...,e_m\}\subseteq E \\ |E'|\leq k}} \ \bigwedge_{1\leq i\leq m} \left(\overline{\mathbf{e}^i}(e_i) \wedge \neg \mathtt{path\text{-}edge\text{-}overlap}(\overline{\mathbf{p}^D}, \overline{\mathbf{e}^i})\right) .\end{aligned} \tag{13}$$

Theorem 3. *Let $E_{fail} \subseteq E$ be a subset of failed edges where $|E_{fail}| \leq k$, and let the vector of variable encodings $\overline{\mathbf{e}^K}$ encode the set E_{fail}. Then $(P, \omega, E_{fail}) \in [\![\mathtt{failover}^k(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}, \overline{\mathbf{e}^K})]\!]$ iff $(P, \omega) \in [\![\mathtt{rsa}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})]\!]$ is a solution to the k-link failover problem with link failures E_{fail}.*

Proof (Proof sketch). "$\Longrightarrow$" Let $(P, \omega, E_{fail}) \in [\![\mathtt{failover}^k(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}, \overline{\mathbf{e}^K})]\!]$ where $|E_{fail}| \leq k$. From Condition 13, we know that $(P, \omega) \in [\![\mathtt{rsa}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})]\!]$ and thereby is a valid solution to the RSA problem. Furthermore, as $\overline{\mathbf{e}^K}$ encodes E_{fail}, we know that for every edge e in E_{fail} that there exists a variable encoding $\overline{\mathbf{e}^i}$ from $\overline{\mathbf{e}^K}$ that encodes e. Additionally, we know that for all e in E_{fail} that $(P, e) \notin [\![\mathtt{path\text{-}edge\text{-}overlap}(\overline{\mathbf{p}^D}, \overline{\mathbf{e}^i})]\!]$ and thus none of the paths assigned to the demands contain any of the failed edges, which means that (P, ω, E_{fail}) is a solution.

"$\Longleftarrow$" Let $(P, \omega) \in [\![\mathtt{rsa}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})]\!]$ be a solution to the k-link failover problem for a set of failed edges $E_{fail} \subseteq E$ where $|E_{fail}| \leq k$, and demands D. By Definition 2, we know that for all $\pi \in P(D)$ that $\pi \cap E_{fail} = \emptyset$. Hence, the BDD $\neg \mathtt{path\text{-}edge\text{-}overlap}(\overline{\mathbf{p}^D}, \overline{\mathbf{e}^i})$ is satisfied for all e. As such, Condition 13 is satisfied and $(P, \omega, E_{fail}) \in [\![\mathtt{failover}^k(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}, \overline{\mathbf{e}^K})]\!]$. □

Using the BDD $\mathit{failover}^k$, we can finally define the BDD $\mathit{path\text{-}pruned\text{-}rsa}$ which encodes all valid solutions given a set of specific link failures E_{fail} where $|E_{fail}| \leq k$

$$\mathit{path\text{-}pruned\text{-}rsa}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}) = \tag{14}$$
$$\exists \overline{\mathbf{e}^K}.\Big(\bigwedge_{e_i \in E_{fail} = \{e_1, e_2, \ldots, e_m\}} \overline{\mathbf{e}^i}(e_i)\Big) \wedge \Big(\bigwedge_{m < j \leq k} \overline{\mathbf{e}^j}(e_{unused})\Big) \wedge \mathit{failover}^k(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}, \overline{\mathbf{e}^K})$$

satisfying that $(P, \omega) \in [\![\mathit{path\text{-}pruned\text{-}rsa}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})]\!]$ iff $(P, \omega) \in [\![\mathit{rsa}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})]\!]$ and $P(d) \cap E_{fail} = \emptyset$ for all $d \in D$. Note that we reuse the name $\mathit{path\text{-}pruned\text{-}rsa}$ from Equation 11 since the BDDs in Equations 11 and 14 are idential for any given failure scenario of size at most k.

5.3 Lightpath-preserving solutions

A property that is often desirable in fast failover protection in a given a failure scenario is that we update the routes and spectrum allocations only for the demands that are affected by some of the failing edges while preserving the remaining lightpaths. This will minimize the number of configuration updates and result in fewer traffic interruptions.

Let (P, ω) be an existing routing and spectrum assignment and let $E_{fail} \subseteq E$ be a given failure scenario. A solution (P', ω') to the failover problem for the set E_{fail} is *lightpath-preserving* if $P(d) \cap E_{fail} = \emptyset$ implies that $P(d) = P'(d)$ and $\omega(d) = \omega'(d)$ for all $d \in D$. We can identify all lightpath-preserving solutions for a given failure scenario E_{fail} as follows:

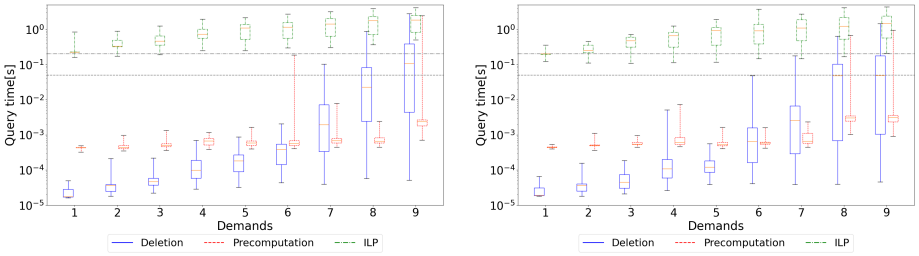
$$\mathit{rsa\text{-}lightpath\text{-}preserving}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}) = \tag{15}$$
$$\mathit{path\text{-}pruned\text{-}rsa}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D}) \wedge \Big(\bigwedge_{\substack{d \in D, \\ P(d) \cap E_{fail} = \emptyset}} \overline{\mathbf{p}^d}(P(d)) \wedge \overline{\mathbf{c}^d}(\omega(d))\Big)$$

where $(P', \omega') \in [\![\mathit{rsa\text{-}lightpath\text{-}preserving}(\overline{\mathbf{p}^D}, \overline{\mathbf{c}^D})]\!]$ iff (P', ω') is a *lightpath-preserving* solution to the given RSA solution (P, ω) for the link failures E_{fail}.

6 Implementation and Evaluation

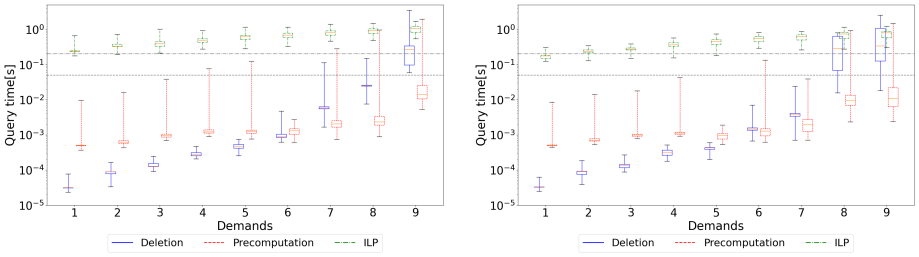
We implemented the BDD encoding of the RSA problem presented in Section 4.2 and the two approaches for solving the k-link failover problem from Section 5 in our open-source tool *ExpectAll*, using the improvements from Section 4.2. The tool is implemented in Python using a Cython wrapper [20] of the library CUDD [52] to perform BDD operations. The source code of *ExpectAll*, including the reproducibility package, is publicly available at [6].

We now evaluate our two BDD approaches to the k-link failover problem against the ILP approach presented in Section 3. The BDD approach using path-deletion pruning is called *deletion*, and the one using precomputation pruning



(a) DT topology (b) Kanto topology

Fig. 4: Query time to find an optimal solution. Lines are shown for 50 ms and 200 ms.

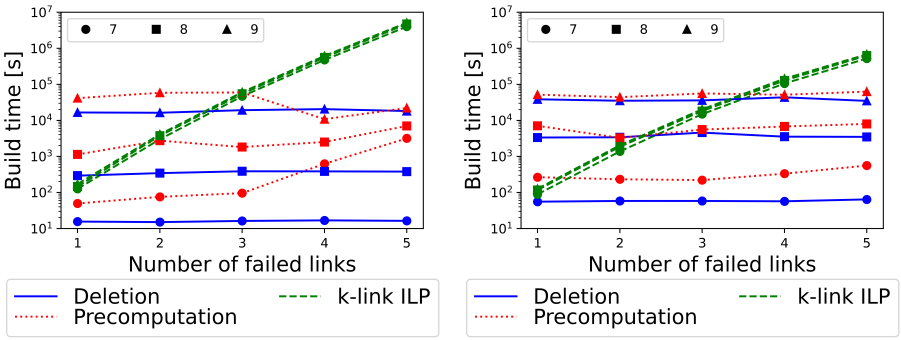


(a) DT topology (b) Kanto topology

Fig. 5: Query time to find an optimal lightpath-preserving solution. Lines are shown for 50 ms and 200 ms.

is called *precomputation*. We compare the query times for finding an optimal solution and an optimal lightpath-preserving solution, by simulating 1000 random k-link failure scenarios using the same experimental setup as described in Section 3. The results are depicted in Figures 4 and 5 for five link failures. We observe that both BDD approaches significantly outperform the ILP model (note the logarithmic scale on the y-axis). The query time of the *deletion* method however increases with the number of demands, exceeding the 200 ms threshold at 8-9 demands. On the other hand, the *precomputation* method has lower variance and maintains average query times consistently below 50 ms, demonstrating its suitability for a reliable, optimal and fast failover recovery. Furthermore, the results in Figure 5 show that while the ILP approach is slightly faster at finding an optimal lightpath-preserving solution compared to a general optimal solution, both our BDD approaches are still significantly faster.

We shall now focus on the time required to compute the BDD representation of all solutions. While a fast (< 50 ms) reaction time to the link failure is high priority, link failures are in general infrequent and we can afford to spend more time (hours) to precompute the BDDs for a given topology and set of demands.



(a) DT topology with 52 links (b) Kanto topology with 36 links

Fig. 6: Build time plots; the three line shapes denote 7,8 and 9 demands

The experiments, presented in Figure 6 for 7, 8 and 9 demands, indicate that as expected the ILP computation time grows exponentially with the number of failed links. In contrast, the build time for the *deletion* method remains nearly constant because the BDD always represents the same number of solutions, rendering the method independent of the number of link failures. The build time of the *precomputation* method increases with the number of link failures, though still remains within acceptable bounds and clearly outperforms the ILP approach (again the y-axis is logarithmic).

The plots also depict a limitation of our method as the precomputation does not yet scale for instances with 10 or more demands. This indicates that our current method is suitable for fast and effective failover protection of a smaller number of critical, high-bandwidth lightpaths that are stable over time. For the remaining, less critical and short-lived demands, we suggest to use e.g. existing heuristic approaches for failover protection at higher spectrum frequency slots, even though optimality is not guaranteed and this approach can lead to possible issues with lack of spectrum slots.

7 Conclusion

We introduced and implemented *ExpectAll*, a novel approach and a tool to efficiently compute and represent *all* valid route and spectrum allocations for a given all-optical network topology. Our approach guarantees optimality and fast reaction time in case of multiple link failures. It outperforms the state-of-the-art approaches based on integer linear programming by orders of magnitude, both in the response time to link failures as well as the precomputation time for three and more concurrently failing links. If we relax the optimality criterium, heuristic approaches for the route and spectrum allocations can be also considered for fast failover protection and we suggest a combination of the heuristic path allocation with *ExpectAll* so that the critical and long-lived lightpath requests

can be dealt with in an optimal way using our method, while the remaining demands can be protected in a less optimal way using the heuristic approaches.

Our experimental evaluation indicates that we can handle up to 9 critical demands in an optimal manner. Even though this is currently the best-scaling technique that guarantees optimality, in the future work we plan to focus on further improving the scalability of the method, e.g. by exploring ideas that can eliminate symmetries in spectrum allocation.

Acknowledgement This work was supported by the Independent Resarch Fund Denmark (DFF) under the project QASNET.

References

1. Akers. Binary decision diagrams. *IEEE Transactions on computers*, 100(6):509–516, 1978.
2. H.R. Andersen. An introduction to binary decision diagrams. *Lecture notes, available online, IT University of Copenhagen*, 5, 1997.
3. K.D.R. Assis, R.C. Almeida Jr, M.J. Reed, A.F. Santos, H.A. Dinarte, D.A.R. Chaves, Haiyuan Li, S. Yan, R. Nejabati, and Dimitra Simeonidou. Protection by diversity in elastic optical networks subject to single link failure. *Optical Fiber Technology*, 75:103208, 2023.
4. P. Athe and Y. Singh. Improved double cycle and link pair methods for two-link failure protection. *Telecommunication Systems*, 74, 05 2020.
5. A. Atlas and A. Zinin. Basic specification for IP fast reroute: Loop-free alternates. *RFC*, 5286:1–31, 2008.
6. G.S. Bruhns, M.P. Hansen, R. Hebsgaard, F.M.W. Hyldgaard, and J. Srba. Reproducibility package for "ExpectAll: A BDD based approach for link failure resilience in elastic optical networks", 2024. Zenodo: `https://doi.org/10.5281/zenodo.14179191`.
7. R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 100(8):677–691, 1986.
8. R. Bryant. Symbolic Boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys*, 24, 03 2003.
9. A. Castro, L. Velasco, J. Comellas, and G. Junyent. On the benefits of multi-path recovery in flexgrid optical networks. *Photonic network communications*, 28:251–263, 2014.
10. S. Chatterjee and S. Pawlowski. All-optical networks. *Commun. ACM*, 42(6):74–83, 1999.
11. M. Chiesa, A. Kamisiński, J. Rak, G. Rétvári, and S. Schmid. A survey of fast recovery mechanisms in the data plane. *TechRxiv*, 2020. `https://doi.org/10.36227/techrxiv.12367508.v2`.
12. M. Chiesa, A. Kamisinski, J. Rak, G. Retvari, and S. Schmid. A survey of fast-recovery mechanisms in packet-switched networks. *IEEE Communications surveys and tutorials*, 23(2):1253–1301, 2021.
13. M. Chiesa, R. Sedar, G. Antichi, M. Borokhovich, A. Kamisinski, G. Nikolaidis, and S. Schmid. PURR: a primitive for reconfigurable fast reroute: hope for the best and program for the worst. In *CoNEXT'19*, pages 1–14. ACM, 2019.

14. I. Chlamtac, A. Ganz, and G. Karmi. Lightpath communications: An approach to high bandwidth optical WAN's. *IEEE transactions on communications*, 40(7):1171–1182, 1992.
15. K. Christodoulopoulos, I. Tomkos, and E.A. Varvarigos. Elastic bandwidth allocation in flexible OFDM-based optical networks. *Journal of Lightwave Technology*, 29(9):1354–1366, 2011.
16. Cisco Systems, Inc. Cisco Annual Internet Report (2018–2023) White Paper. `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html`, 2020.
17. G. Corfield. British airways' latest total inability to support upwardness of planes caused by amadeus system outage, 2018. `https://www.theregister.co.uk/2018/07/19/amadeus_british_airways_outage_load_sheet`.
18. S. Eachempati, V. Saripalli, N. Vijaykrishnan, and S. Datta. Reconfigurable BDD based quantum circuits. In *2008 IEEE International Symposium on Nanoscale Architectures*, pages 61–67. IEEE, 2008.
19. Metro Ethernet. Technical specification MEF 2 requirements and framework for ethernet service protection in metro ethernet networks. Technical report, Metro Ethernet, 2004. `https://www.mef.net/resources/mef-2-requirements-and-framework-for-ethernet-service-protection/`.
20. I. Filippidis, S. Haesaert, S.C. Livingston, and M. Wenzel. Python package tulip-control/dd, 2024. `https://github.com/tulip-control/dd?tab=readme-ov-file`.
21. K.-T. Foerster, M. Parham, M. Chiesa, and S. Schmid. TI-MFA: Keep calm and reroute segments fast. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 415–420. IEEE, 2018.
22. P. François, C. Filsfils, A. Bashandy, B. Decraene, and S. Litkowski. Topology independent fast reroute using segment routing. *Network Working Group*, 2014. `https://datatracker.ietf.org/doc/html/draft-francois-spring-segment-routing-ti-lfa-02`.
23. T. Gao, W. Zou, X. Li, B. Guo, S. Huang, and B. Mukherjee. Distributed sublight-tree based multicast provisioning with shared protection in elastic optical datacenter networks. *Optical Switching and Networking*, 31:39–51, 2019.
24. C. Gibbs. ATT's 911 outage result of mistakes made by ATT, FCC's Pai says, 2017. `https://www.fiercewireless.com/wireless/at-t-s-911-outage-result-mistakes-made-by-at-t-fcc-s-pai-says`.
25. Ph. Gill, N. Jain, and N. Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. In *Proceedings of the ACM SIGCOMM 2011 Conference*, pages 350–361, 2011.
26. Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. `https://www.gurobi.com`.
27. C. Gyorgyi, K.G. Larsen, S. Schmid, and J. Srba. SyPer: synthesis of perfectly resilient local fast re-routing rules for highly dependable networks. In *IEEE International Conference on Computer Communications (INFOCOM'24)*, pages 2398–2407. IEEE, 2024.
28. C. Gyorgyi, K.G. Larsen, S. Schmid, and J. Srba. SyRep: Efficient synthesis and repair of fast re-route forwarding tables for resilient networks. In *Proceedings of the 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'24)*, pages 483–494. IEEE, 2024.
29. D.T. Hai and K.M. Hoang. An efficient genetic algorithm approach for solving routing and spectrum assignment problem. In *2017 international conference on re-*

cent advances in signal processing, telecommunications & computing (SigTelCom), pages 187–192. IEEE, 2017.
30. D.T. Hai, M. Morvan, and Ph. Gravey. Combining heuristic and exact approaches for solving the routing and spectrum assignment problem. *IET Optoelectronics*, 12(2):65–72, 2018.
31. M. Jinno, H. Takara, and B. Kozicki. Dynamic optical mesh networks: Drivers, challenges and solutions for the future. In *35th European Conference on Optical Communication*, pages 1–4. IEEE, 2009.
32. N.S. Johansen, L.B. Kaer, A.L. Madsen, K.O. Nielsen, S. Schmid, J. Srba, and R.G. Tollund. FBR: Dynamic memory-aware fast rerouting. In *IEEE Global Internet (GI) Symposium 2022*, pages 55–60. IEEE, 2022.
33. G.E. Keiser. A review of WDM technology and applications. *Optical Fiber Technology*, 5(1):3–39, 1999.
34. M. Klinkowski and K. Walkowiak. Routing and spectrum assignment in spectrum sliced elastic optical path network. *IEEE Communications Letters*, 15:884–886, 08 2011.
35. S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.
36. K. Kubota, Y. Tanigawa, Y. Hirota, and H. Tode. Crosstalk-aware resource allocation based on optical path adjacency and crosstalk budget for space division multiplexing elastic optical networks. *IEICE Transactions on Communications*, 107(1):27–38, 2024.
37. K.G. Larsen, A. Mariegaard, S. Schmid, and J. Srba. AllSynth: Transiently correct network update synthesis accounting for operator preferences. In *International Symposium on Theoretical Aspects of Software Engineering*, pages 344–362. Springer, 2022.
38. C. Y. Lee. Representation of switching circuits by binary-decision programs. *The Bell System Technical Journal*, 38(4):985–999, 1959.
39. F. Lezama, A. Martínez-Herrera, G. Castanon, C. Del Valle Soto, A. Sarmiento, and E. Munoz de Cote. Solving routing and spectrum allocation problems in flexgrid optical networks using pre-computing strategies. *Photonic Network Communications*, 41:1–19, 02 2021.
40. X. Li, S. Huang, J. Zhang, Y. Zhao, W. Gu, and Y. Wang. Analysis and modeling of k-regular and k-connected protection structure in ultra-high capacity optical networks. *China Communications*, 12:106–119, 03 2015.
41. G. Markovic. Routing and spectrum allocation in elastic optical networks using bee colony optimization. *Photonic Network Communications*, 34, 12 2017.
42. Y. Miyagawa, Y. Watanabe, M. Shigeno, K. Ishii, A. Takefusa, and A. Yoshise. Bounds for two static optimization problems on routing and spectrum allocation of anycasting. *Optical switching and networking*, 31:144–161, 2019.
43. B. Mukherjee. *Optical communication networks*. McGraw-Hill, 1997.
44. P. Pan, G. Swallow, and A. Atlas. Fast reroute extensions to RSVP-TE for LSP tunnels. *RFC*, 4090:1–38, 2005.
45. J. Papan, P. Segec, M. Moravcik, M. Kontsek, L. Mikus, and J. Uramova. Overview of IP fast reroute solutions. In *ICETA*, pages 417–424. IEEE, 2018.
46. R. Ramaswami and K.N. Sivarajan. Routing and wavelength assignment in all-optical networks. *IEEE/ACM Transactions on networking*, 3(5):489–500, 1995.
47. A. Rauchenecker and R. Wille. An efficient physical design of fully-testable BDD-based circuits. In *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, pages 6–11. IEEE, 2017.

48. M. Roughan, A. Greenberg, Ch. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang. Experience in measuring backbone traffic variability: models, metrics, measurements and meaning. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment (IMW'02)*, page 91–92. ACM, 2002.
49. T. Sakano, Y. Tsukishima, H. Hasegawa, T. Tsuritani, Y. Hirota, S. Arakawa, and H. Tode. A study on a photonic network model based on the regional characteristics of Japan. *PN2013-1*, 113(91):1–6, 2013.
50. G. Shen, Y. Wei, and S.K. Bose. Optimal design for shared backup path protected elastic optical networks under single-link failure. *Journal of Optical Communications and Networking*, 6(7):649–659, 2014.
51. N.K. Singhal, C. Ou, and B. Mukherjee. Cross-sharing vs. self-sharing trees for protecting multicast sessions in mesh networks. *Computer Networks*, 50(2):200–206, 2006. Optical Networks.
52. F. Somenzi. CUDD: CU decision diagram package release 2.4.1, 2024. `https://web.mit.edu/sage/export/tmp/y/usr/share/doc/polybori/cudd/cuddIntro.html`.
53. Switch Specification 1.3.1. OpenFlow. In *Open Networking Foundation*, 2013. `https://bit.ly/2VjOO77`.
54. T. Tanaka and M. Shimoda. Pre-and post-processing techniques for reinforcement-learning-based routing and spectrum assignment in elastic optical networks. *Journal of Optical Communications and Networking*, 15(12):1019–1029, 2023.
55. I. van Duijn, P.G. Jensen, J.S. Jensen, T.B. Krøgh, J.S. Madsen, S. Schmid, J. Srba, and M.T. Thorgersen. Automata-theoretic approach to verification of MPLS networks under link failures. *IEEE/ACM Transactions on Networking*, 30(2):766–781, 2022.
56. L. Velasco Esteban, M. Klinkowski, M. Ruiz, and J. Comellas. Modeling the routing and spectrum allocation problem for flexgrid optical networks. *Photonic Network Communication*, 24:177–186, 01 2013.
57. J. Wang, M. Shigeno, and Q. Wu. ILP models and improved methods for the problem of routing and spectrum allocation. *Optical Switching and Networking*, 45:100675, 2022.
58. Y. Wang, X. Cao, and Y. Pan. A study of the routing and spectrum allocation in spectrum-sliced elastic optical path networks. In *Proceedings IEEE INFOCOM'11*, pages 1503–1511. IEEE, 2011.
59. Wikipedia. List of cities in germany by population. `https://en.wikipedia.org/wiki/List_of_cities_in_Germany_by_population`.
60. L. Xu, Y.-C. Huang, Y. Xue, and X. Hu. Deep reinforcement learning-based routing and spectrum assignment of EONs by exploiting GCN and RNN for feature extraction. *Journal of Lightwave Technology*, 40(15):4945–4955, 2022.
61. H. Zang, J.P. Jue, and B. Mukherjee. A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. *Optical networks magazine*, 1(1):47–60, 2000.
62. J. Zhang, P. Miao, and F. Zhang. On optimal routing and spectrum allocation in elastic optical networks. In *2nd International Conference on Big Data, Information and Computer Network (BDICN)'23*, pages 284–287. IEEE, 2023.