Survivable Virtual Network Embedding in Transport Networks

by

Nashid Shahriar

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2020

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Chadi M. Assi Professor Gina Cody School of Engineering Concordia University
Supervisor:	Raouf Boutaba Professor David R. Cheriton School of Computer Science University of Waterloo
Internal Member:	Samer Al-Kiswany Assistant Professor David R. Cheriton School of Computer Science University of Waterloo
Internal Member:	Yaoliang Yu Assistant Professor David R. Cheriton School of Computer Science University of Waterloo
Internal-External Member:	Pin-Han Ho Professor Department of Electrical and Computer Engineering University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This dissertation includes first authored peer reviewed materials that have appeared in conference and journal proceedings published by the Institute of Electrical and Electronics Engineers (IEEE).

The IEEE's policy on reuse of published materials in a dissertation is as follows: "The IEEE does not require individuals working on a thesis to obtain a formal reuse license" The following list serves as a declaration of the Versions of Record for works included in this dissertation:

Portions of Chapter 2:

Nashid Shahriar, Reaz Ahmed, Shihabur Rahman Chowdhury, Md Mashrur Alam Khan, Raouf Boutaba, Jeebak Mitra, and Feng Zeng. "Virtual Network Embedding With Guaranteed Connectivity Under Multiple Substrate Link Failures," in IEEE Transactions on Communications, vol. 68, no. 2, pp. 1025-1043, Feb. 2020,
doi: 10.1109/TCOMM.2019.2954410.

Portions of Chapter 3:

Nashid Shahriar, Shihabur Rahman Chowdhury, Reaz Ahmed, Aimal Khan, Siavash Fathi, Raouf Boutaba, Jeebak Mitra, and Liu Liu. "Virtual Network Survivability Through Joint Spare Capacity Allocation and Embedding," in IEEE Journal on Selected Areas in Communications, vol. 36, no. 3, pp. 502-518, March 2018, doi: 10.1109/JSAC.2018.2815430.

Portions of Chapter 4:

Nashid Shahriar, Reaz Ahmed, Shihabur Rahman Chowdhury, Aimal Khan, Raouf Boutaba, and Jeebak Mitra. "Generalized Recovery From Node Failure in Virtual Network Embedding," in IEEE Transactions on Network and Service Management, vol. 14, no. 2, pp. 261-274, June 2017, doi: 10.1109/TNSM.2017.2693404.

Portions of Chapter 5:

Nashid Shahriar, Sepehr Taeb, Shihabur Rahman Chowdhury, Mubeen Zulfiqar, Massimo Tornatore, Raouf Boutaba, Jeebak Mitra, and Mahdi Hemmati. "Reliable Slicing of 5G Transport Networks with Bandwidth Squeezing and Multi-path Provisioning," in IEEE Transactions on Network and Service Management,
doi: 10.1109/TNSM.2020.2992442.

Abstract

Network Virtualization (NV) is perceived as an enabling technology for the future Internet and the 5th Generation (5G) of mobile networks. It is becoming increasingly difficult to keep up with emerging applications' Quality of Service (QoS) requirements in an ossified Internet. NV addresses the current Internet's ossification problem by allowing the co-existence of multiple Virtual Networks (VNs), each customized to a specific purpose on the shared Internet. NV also facilitates a new business model, namely, Network-as-a-Service (NaaS), which provides a separation between applications and services, and the networks supporting them. 5G mobile network operators have adopted the NaaS model to partition their physical network resources into multiple VNs (also called network slices) and lease them to service providers. Service providers use the leased VNs to offer customized services satisfying specific QoS requirements without any investment in deploying and managing a physical network infrastructure.

The benefits of NV come at additional resource management challenges. A fundamental problem in NV is to efficiently map the virtual nodes and virtual links of a VN to physical nodes and paths, respectively, known as the Virtual Network Embedding (VNE) problem. A VNE that can survive physical resource failures is known as the survivable VNE (SVNE) problem, and has received significant attention recently. In this thesis, we address variants of the SVNE problem with different bandwidth and reliability requirements for transport networks. Specifically, the thesis includes four main contributions. First, a connectivity-aware VNE approach that ensures VN connectivity without bandwidth guarantee in the face of multiple link failures. Second, a joint spare capacity allocation and VNE scheme that provides bandwidth guarantee against link failures by augmenting VNs with necessary spare capacity. Third, a generalized recovery mechanism to re-embed the VNs that are impacted by a physical node failure. Fourth, a reliable VNE scheme with dedicated protection that allows tuning of available bandwidth of a VN during a physical link failure. We show the effectiveness of the proposed SVNE schemes through extensive simulations. We believe that the thesis can set the stage for further research specially in the area of automated failure management for next generation networks.

Acknowledgements

First and foremost, I thank Allah, the Exalted in Might, the Wisest, and the Omniscient, for bringing me into existence and enabling me to complete this work.

I express my warmest gratitude to my supervisor, Professor Raouf Boutaba, for supporting me with a continuous stream of intellectual, moral, and financial assistance. He has always encouraged me to pursue my ideas and provided invaluable feedback to improve the quality of my research. I am especially thankful for his honest feedback and mentorship that has developed me as a researcher and a person.

I want to extend my gratitude and acknowledgments to Professor Chadi M. Assi, Professor Samer Al-Kiswany, Professor Yaoliang Yu, and Professor Pin-Han Ho for their insightful comments and constructive criticisms on this thesis. I am thankful to my colleagues and mentors, Dr. Reaz Ahmed, Dr. Massimo Tornatore, Dr. Jeebak Mitra, Shihabur Rahman Chowdhury, and Sepehr Taeb for their support, enlightening discussions, and a lot of cheerful moments.

I owe my deepest gratitude to my beloved wife Shahaly Zabin and my son Sharfan Ahmed for their patience and support during the time of hardship. I am truly grateful to my mother, parents-in-law, and sister for constantly inspiring me in the quest of knowledge and for securing a paved way for my career plans. Without the support and love from my family members, it would be impossible for me to complete this work.

I thank all my teachers for their inspiration, wisdom and endless efforts. Last but not least, I thank everyone who has contributed to this thesis, directly or indirectly.

Dedication

To my father.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Network Virtualization (NV) is an enabling technology for the future Internet and the 5th Generation (5G) of mobile networks [45, 10]. The current Internet suffers from "ossification" as the Internet's size and rigidity make it very hard to adopt new networking technologies [162]. For example, transition from Internet Protocol version 4 (IPv4) to IPv6 has started more than a decade ago, however, IPv6 adoption rate is still significantly low as reported by major service providers (*e.g.*, less than 30% of Google users adopted IPv6 [2]). It is becoming increasingly cumbersome to keep up with emerging applications' Quality of Service (QoS) requirements in terms of bandwidth, reliability, throughput, and latency in an ossified Internet. NV solves the ossification problem by allowing the co-existence of multiple virtual networks, each of which is customized to a specific purpose (*e.g.*, a virtual network with low-power sensors using IPv4), on the shared Internet.

5G networks are on the horizon with the promise of revolutionizing the communication landscape and our way of living. 5G will provide the communication infrastructure to realize emerging technologies, including smart city, industry automation, e-health, intelligent transportation, among others [57]. These technologies will enable a wide variety of services and applications with diverse QoS requirements in terms of bandwidth, latency, jitter, reliability, energy efficiency, and mobility [116]. For instance, telehealth services need the highest level of reliability with a bandwidth guarantee, whereas real-time monitoring and control services require data to be delivered satisfying a stringent end-to-end (E2E) latency budget. Satisfying diverse requirements imposed by various applications and services poses a challenge to 5G network operators. NV has been proposed to cope with this challenge [33]. 5G mobile network operators will use NV technologies to partition their physical network resources into multiple virtual networks (also called network slices) tailored to specific QoS requirements [121].

NV also facilitates a new business model, namely, Network-as-a-Service (NaaS), which provides a separation between applications and services, and the networks supporting them [33]. Infrastructure Providers (InPs) and mobile network operators have adopted the NaaS model to deploy customized virtual networks (network slices) and lease them to Service Providers (SPs) [121]. Service providers use the leased virtual networks to offer services with diverse QoS requirements (*e.g.*, enhanced mobile broadband, ultra-reliable and low latency communication, and massive machine connectivity) without any investment in deploying and managing a physical infrastructure.

The benefits of NV comes with additional resource management challenges for infrastructure providers as they have to keep virtual networks up and running throughout their life-spans. A virtual network (VN), irrespective of being a slice of access, core, or transport network, usually consists of virtual nodes (e.g., virtual machines and virtual switches) and virtual links representing paths in the physical network, forming a virtual topology. For instance, a virtual network in the Internet Protocol (IP) layer comprises of routers/switches and overlay IP links connecting them, whereas a virtual network in the optical layer connects optical transceivers through a lightpath (*i.e.*, a circuit of fiber links) [90]. In 5G, a network slice is an end-to-end virtual network that spans multiple technological and administrative network segments (*e.g.,* wireless radio, access/core transport networks, edge, and central Data Centers (DCs)). A fundamental problem in NV is to allocate physical resources to virtual networks, which is known as the Virtual Network Embedding (VNE) problem [59]. The VNE problem respectively maps virtual nodes and links of a virtual network to physical nodes and paths (a sequence of links) in a physical network, while satisfying resource budgets and achieving specific objectives such as minimizing resource usage [46].

Virtual networks cannot function properly in the face of network device or link failures that are inevitable in everyday operations [3, 65, 117]. Such failures, if not treated properly, may disrupt the services hosted by virtual networks, incurring high penalties in terms of revenue losses and SLA violations [67]. For example, every minute of outage can cost a network provider up to USD 5,600, leading to well over USD 300K per hour [1]. While accepting the possibility of failures, infrastructure providers have to ensure different levels of availability (e.g., 99.999%, 99.9999%, and so on) for their virtual networks as part of SLAs [72]. Survivability mechanisms can greatly reduce the impact of failures, while minimizing downtime and upholding the reputation of a network provider [78]. In this context, a VN embedding that can survive substrate failures is known as the survivable VNE (SVNE) [132], and has received significant attention from the research community.

This dissertation addresses variants of the SVNE problem with bandwidth and reliability requirements for transport networks that provide data transmission services over large

geographical (metro, regional, or national) areas. The considered transport technologies are: Transport Software Defined Networks (T-SDNs) [13] and Elastic Optical Networks (EONs) [28], expected to meet the demanding requirements of 5G networks. In these networks, failures (*e.g.*, device outage or fiber-cut) can result in significant amount of data loss [117, 65]. Existing SVNE strategies advocate for keeping failure management tasks transparent to the service provider and support a limited set of QoS requirements [78]. In contrast, this dissertation argues for delegating failure management responsibilities to a virtual network operator. Offloading failure management to a virtual network has two benefits: i) enabling a variety of QoS requirements through different survivability models; ii) offering more opportunities to minimize resource consumption. This dissertation presents four different survivability models to enable failure management at the virtual network layer as opposed to the physical network layer. Specifically, this dissertation makes the following contributions:

- Guaranteeing virtual network connectivity against multiple link failures in T-SDNs.
- Jointly optimizing spare capacity allocation and survivable virtual network embedding to guarantee bandwidth in the presence of multiple link failures in T-SDNs.
- Re-embedding a batch of virtual networks to recover from a node failure T-SDNs.
- Reliable slicing of EONs to ensure fast fail-over of virtual networks against a link failure in EONs.

A more detailed overview of these contributions is provided in the remainder of this chapter.

1.1 Connectivity-aware Virtual Network Embedding

A critical challenge in SVNE is to ensure **Co***nnectivty-aware* **Vi***rtual* **N***etwork* **E***mbedding* (*CoViNE*) [141, 142]. The goal of *CoViNE* is to find a VN embedding that remains connected (without any bandwidth guarantee) in the presence of multiple substrate link failures in T-SDNs. Guaranteeing connectivity in the VN embedding will incur less resource overhead and reduced cost of leasing resources for a VN, however, providing a weaker form of survivability. This survivability model is well-suited for VNs that carry best-effort traffic (*e.g.*, file transfer and email communication) and can tolerate disruption during failure restoration. Upon failures, the affected VN traffic can be rerouted to alternate paths following any predefined policy, *e.g.*, customer priority. *CoViNE*'s survivability model also

allows to delegate failure handling responsibility to an SP, which can then use a Software Defined Network (SDN) controller to employ their own network design and restoration techniques instead of simply relying on the InP [154, 91, 74]. Our specific contributions in this work are:

- *CoViNE*, an alternate survivability model for VNE, which requires significantly less backup resources than traditional survivability approaches in the SVNE literature.
- Three novel solutions to *CoViNE* to guarantee VN connectivity under multiple substrate link failures. We formulated *CoViNE* as an Integer Linear Program (ILP) for computing the optimal solution. To address the computational complexity of the ILP, we devise two heuristic algorithms by decomposing *CoViNE* into multiple sub-problems and solving them sequentially.

We performed extensive simulations to evaluate the optimality and scalability of the proposed solutions under single and double substrate link failures. We restrict our simulations to one and two link failures since the probability of more than two simultaneous link failures is extremely low [117, 65]. We also compare our solutions with a VNE approach that does not guarantee VN connectivity upon failures. Through our simulation study, we show that virtual network connectivity is ensured against single link failures with about 25% additional resources, compared to a VNE that does not guarantee connectivity.

1.2 Joint Spare Capacity Allocation and Virtual Network Embedding

Virtual networks for augmented/virtual reality or telehealth applications may need bandwidth guarantee even in the presence of failures in T-SDNs. One way of providing such guarantee is to allocate spare bandwidth on virtual links (as opposed to doing so on the physical network) so that all the traffic impacted by a failure can be rerouted within the virtual network [144, 145]. With this type of survivability, InPs can offload failure management tasks to SPs by augmenting VNs with sufficient spare capacity for backup and embedding the VNs in a way that primary and backup VN resources are not affected by the same substrate resource failure. When a substrate resource fails, it is the SP's responsibility to reroute the affected traffic to the pre-allocated backup resources within the VN.

Independently addressing spare bandwidth allocation and VNE may lead to sub-optimal solutions. In this work, we study the joint optimization problem of computing spare bandwidth allocation and VNE with the objective of guaranteeing VN survivability under multiple substrate link failures and minimizing resource usage in the SN. We start with single link failure scenario and then extend the solution to survive multiple link failures. Specifically, we make the following contributions:

- We formulate a joint optimization model using a Quadratic Integer Program (QIP) to optimally solve spare capacity allocation and survivable VN embedding simultaneously. We transform the QIP into an Integer Linear Program (ILP) without sacrificing its optimality. We provide two more ILP formulations for solving two extreme cases of spare capacity sharing. We present a mathematical analysis that dictates how the topological properties of the SN affect the level of spare capacity sharing.
- The ILP formulations for the joint optimization problem are not scalable to large problem instances. Hence, we devise an efficient heuristic algorithm to tackle the computational complexity of the ILP-based solutions. The algorithm leverages a novel spare bandwidth sharing model to estimate the spare capacity and computes embedding based on the estimated spare capacity. In the final step, the algorithm re-optimizes spare capacity allocation based on the final embedding information.

We perform simulations to evaluate our solutions for single and double link failures. We also perform a quantitative comparison between the SVNE with VN level protection and the traditional SVNE with physical network level protection. Our evaluation shows that our solution decreases resource usage by 30% compared to traditional SVNE solutions that rely on the physical network to perform failure management.

1.3 Recovery from Node Failure in Virtual Network Embedding

In this work, we study the problem of **Re**covering from a **No**de failure in **V**irtu**a**l Ne**t**work **E**mbedding (*ReNoVatE*) [143, 140]. *ReNoVatE* takes a batch of VN failures resulting from a single substrate node failure, and produces alternate embeddings for the failed virtual nodes and links. The impact of a substrate node failure is more drastic than that of a link failure since a node failure affects embedding of all the virtual nodes and links of all

VNs passing through the failed node. Preallocating backup resources for multiple failures resulting from a substrate node failure can be extremely expensive [82, 177]. Instead, an SP may prefer to reactively re-embed the failed part of its VN to avoid the huge cost of preallocated backup resources in a failure-prone SN. Such reactive approaches can be adopted by a virtual network whose traffic can tolerate non-negligible service disruptions (*e.g.*, Internet of Things (IoT) traffic for non-realtime services).

To recover from a substrate node failure, the embedding of all affected virtual nodes and links of all impacted VNs should be re-embedded on non-failed components of the SN. The combinatorial number of possibilities of alternate embeddings of the failed virtual nodes and links of the VNs makes the task of finding the most efficient recovery both non-trivial and intractable. Furthermore, any recovery approach should not cause, ideally, any service disruption for the unaffected parts of the VNs. We take into account these issues to design a generalized recovery approach that can achieve various objectives such as fair treatment on the failed VNs, partial treatment based on priority, and so on. Our specific contributions in this work are:

- We propose a generalized recovery approach that allows partial recovery of an affected VN, while adhering to SLA requirements. To demonstrate the versatility of our approach, we investigate two different recovery models. The first one is a fair recovery model (FRM) that maximizes the number of recovered virtual links across all the affected VNs, while minimizing total bandwidth required for recovery. The second model is a priority-based recovery model (PRM) that seeks to prioritize the recovery of affected VNs based on some predefined requirements and minimize total bandwidth needed for recovery.

- We formulate *ReNoVatE* as an ILP based optimization model. Since the optimization model cannot scale to large instances of the problem, we devise an efficient heuristic algorithm to find satisfactory solutions within prescribed time limits.

We evaluate our solutions through extensive simulations and compare them with the most related state-of-the-art proposal in the literature [25]. Our evaluation results demonstrate that our heuristic algorithm performs close to the ILP based optimization model and outperforms the state-of-the-art solution, in terms of i) number of recovered virtual links, ii) cost of recovery, and iii) execution time.

1.4 Reliable Slicing of Elastic Optical Networks

The SVNE problem in EONs is compounded by a number of tunable transmission parameters (*e.g.*, modulation format, baud rate, and error correction codes) and a larger solution space for spectrum allocation. Compared to T-SDNs, EONs have additional constraints imposed by the physical characteristics of optical devices and the properties of light. Furthermore, lightpaths in EONs carry huge volumes of data, and hence, even a short-lived outage can cause a significant traffic loss for virtual networks, necessitating a fast failover (*e.g.*, within 50 milliseconds [133]) capability. To meet this requirement, dedicated protection is the appropriate survivability option, although it incurs a 100% resource overhead that remains unused most of the time. To minimize resource footprint of dedicated protection, we leverage two techniques: bandwidth squeezing rate (BSR) that allows a virtual network operator to tune the amount of available bandwidth in case of failures and multi-path provisioning with demand splitting [148, 146]. These two techniques allow to significantly reduce spectrum usage for providing dedicated protection, especially in the case of fully-flexible EONs. Specifically, we make the following contributions:

- We address the SVNE problem over EONs by capturing the full flexibility of an EON in terms of finer-grained spectrum allocation, adapting modulation format and Forward Error Correction (FEC) overhead for rightsize resource allocation. To minimize resource overhead of dedicated protection based SVNE, we leverage BSR along with multi-path provisioning using demand splitting.
- We develop a mathematical model to optimally solve SVNE on EON by jointly considering all the flexible transmission parameters, while minimizing the total spectrum usage. To the best of our knowledge, this is the first optimization model capturing flexibility in all the transmission parameters of an EON. Given the NP-hardness of the optimal solution, we propose a heuristic algorithm to solve larger instances of the problem.

We perform simulations using realistic network topologies, which provide valuable insight into how different levels of BSR and path diversity in the substrate EON's can impact the extent of backup resource savings for dedicated protection. We also analyze the steady state behavior of our heuristic solution using a discrete event simulator. Our evaluation shows that by using multi-path provisioning, it is possible to guarantee up to 40% of the requested bandwidth of a VN during failure (*i.e.*, $BSR \leq 40\%$) while using as low as 10% additional spectrum resources. Consequently, VN blocking ratio for $BSR \leq 40\%$ remains very similar to that of the case with no backup.

1.5 Thesis Organization

This dissertation proceeds as follows to detail our contributions. In Chapter 2, we present *CoViNE*. We discuss the work on joint spare capacity allocation and virtual network embedding in Chapter 3. Next, we present *ReNoVatE* and reliable slicing of elastic optical networks in Chapter 4 and Chapter 5, respectively. Finally, concluding remarks are presented in Chapter 6 including a summary of the contributions of this thesis and a discussion of potential future research directions.

Chapter 2

Connectivity-aware Virtual Network Embedding

NV operates in a dynamic environment where substrate resources may fail and multiple concurrent failures is not a rare event [85]. Surviving failures is of paramount importance, since a single failure in an SN may result in multiple failures in the embedded VNs. Finding a VN embedding that can survive failures in an SN is known as the Survivable Virtual Network Embedding (SVNE) problem [131]. The majority of the works on SVNE focus on link failures, as they occur more frequently than node failures [117]. SVNE approaches, in general, allocate redundant resources for each (or selected) virtual link(s) and node(s), either pro-actively while computing the embedding or reactively after a failure occurs [78]. Traditionally, proactive SVNE approaches focus on guaranteeing virtual link demand in the presence of failure(s). These approaches assume that InPs handle substrate failures by provisioning redundant backup resources to guarantee virtual links' bandwidth in the event of failures, which in turn incurs additional cost to SPs. Backup resource requirement can increase substantially in a multiple (*e.g.*, k) link failure scenario because of the combinatorial number of k link failure possibilities.

In this chapter, we focus on a different form of survivability than traditional proactive SVNE, namely **Co***nnectivty-aware* **Vi***rtual* **N***etwork* **E***mbedding* (*CoViNE*). Our goal is to find a VN embedding that remains connected (without any bandwidth guarantee) in the presence of multiple substrate link failures. Guaranteeing connectivity in the VN embedding will incur less resource overhead and reduced cost of leasing resources for a VN, however, providing a weaker form of survivability. This survivability model is well-suited for VNs that carry best-effort traffic and can tolerate disruption during failure restoration. Upon failures, the affected VN traffic can be rerouted to alternate paths following any

predefined policy, *e.g.*, customer priority. This survivability model also allows to delegate failure handling responsibility to an SP, which can then use a Software Defined Network (SDN) controller to employ their own network design and restoration techniques instead of simply relying on the InP [154, 91, 74].

Although our focus is NV, *CoViNE* is equally applicable to IP-over-Wavelength-division multiplexing (WDM) networks. The problem of ensuring IP network connectivity in the presence of underlying WDM link failure(s) is known as *link survivable mapping*. Two variations of the link survivability problem have been studied in IP-over-WDM literature [109]: i) *weakly link survivable mapping* (WLSM) ensures only IP-layer connectivity; ii) *strongly link survivable* mapping guarantees both connectivity and bandwidth of the failed IP link(s) against WDM link failures. However, a major difference between VNE and mapping in IP-over-WDM networks is that IP routers are attached to fixed locations in the latter, whereas the placement of virtual nodes is an outcome of VNE algorithm in the former. Therefore, solutions for IP-over-WDM networks such as [104, 156, 182] cannot be directly applied to *CoViNE*. As a matter of fact, WLSM problem is merely a special case of *CoViNE*.

Solving *CoViNE* under k substrate link failures requires satisfying the following necessary and sufficient condition [183]: at least one link in each edge-cut of a VN must remain connected after any k substrate link failures. Given that there can be an exponential number of edge-cuts in a VN and also the combinatorial number of k substrate link failure possibilities, the aforementioned condition becomes impractical to be satisfied even for small substrate networks [183]. Alternatively, the same connectivity guarantee of a VN under k substrate link failures can be achieved by the following two necessary conditions: i) the VN must be $k+1$ edge connected, and ii) at least $k+1$ edge-disjoint paths must exist between every pair of virtual nodes in the embedding of the VN on the SN. The first condition can be satisfied by augmenting the VN with additional virtual links if needed [156, 103]. However, the number of augmented virtual links should be minimized since these augmented links consume substrate resources during VNE. A naive way to satisfy the second condition is to embed all the virtual links of a $k+1$ edge connected VN on disjoint paths in the SN at the cost of increased resource requirements for embedding virtual links [97]. However, as we demonstrate in Section 2.2.4, not all virtual links need to be embedded on disjoint substrate paths to satisfy the second condition. Therefore, a sought-after solution to *CoViNE* should simultaneously optimize the number of augmented virtual links and the disjointness constraints in order to minimize the cost of VN embedding.

Optimally solving *CoViNE* for k substrate link failures is intractable since it entails to jointly optimize VN augmentation, disjointness constraints computation, and embedding

of the augmented VN while satisfying the disjointness and capacity constraints. As each of these problems, when solved independently, is either NP-complete [168] or NP-Hard [97, 185], addressing them simultaneously exacerbates the complexity of *CoViNE*. This challenging problem has not been well studied in the NV literature, although WLSM problem (a special case of *CoViNE*) has received significant attention in the IP-over-WDM literature. The majority of the solutions to WLSM have overlooked the critical step of augmentation, assuming that input VNs have the necessary edge-connectivity [120, 158, 159, 183], which is not always the case. In addition, a significant body of existing literature on WLSM focus on single substrate link failure [182, 120, 158]. Solutions for multiple substrate link failures either fall short in dealing with arbitrary VN topologies [103], or consider only a special case, *i.e.*, Shared Risk Link Group (SRLG) failures [159, 183]. The authors in [156] address a WLSM problem that considers both augmentation and multiple link failures. However, the heuristic solution in [156] requires a large number of virtual links to be embedded disjointly, possibly imposing an unsatisfiable number of disjointness constraints. To overcome these limitations, in this chapter, we present novel solutions to *CoViNE* that can embed an arbitrary VN topology on an SN, while guaranteeing VN connectivity against k substrate link failures and minimizing substrate resource consumption. Our solutions, if needed, augment a VN with minimal number of virtual links and preserve the topological structure of the VN to remain transparent to the SP operating the VN. Specifically, we make the following contributions, which build on our earlier study presented in [141].

First, we present *CoViNE*, an alternate survivability model for VNE, which embeds a VN on an SN subject to the constraint that the VN remains connected under k substrate link failures, *i.e.*, at least one working path exists between every pair of virtual nodes when up to k substrate links fail. While doing so, *CoViNE* minimizes the bandwidth provisioning cost in the SN. The survivability model proposed by *CoViNE* significantly reduces backup resource requirement compared to traditional survivability approaches in the SVNE literature.

Second, we propose three novel solutions to *CoViNE*:

- ***CoViNE-opt***: An Integer Linear Program (ILP) formulation that jointly optimizes VN augmentation, disjointness constraints computation, and embedding of the augmented VN to optimally solve *CoViNE*. *CoViNE-opt* has an exponential number of variables and constraints that severely limits its scalability. To scale to larger problem instances, we decompose *CoViNE* into three sub-problems: (i) augmenting the VN with zero or more virtual links to make it $k+1$-edge connected; (ii) computing the set of virtual links to be embedded disjointly for ensuring connectivity against k substrate link failures; and (iii) embedding the VN while satisfying the aforemen-

tioned disjointness constraints. The following two approaches (*i.e.*, *CoViNE-ILP* and *CoViNE-fast*) sequentially solve all the sub-problems of *CoViNE* in a more scalable manner, however, without guaranteeing an optimal solution.

- ***CoViNE-ILP***: Employs a heuristic that solves sub-problems (i) and (ii) in polynomial time. This heuristic leverages *conflicting set* abstraction resulting from a theoretical analysis of *CoViNE*. The *conflicting set* abstraction allows to generate a polynomial number of variables and constraints to be used by an ILP for solving sub-problem (iii). The complexity of this ILP limits its applicability to substrate networks of few hundred nodes.

- ***CoViNE-fast***: Uses heuristics for all three sub-problems of *CoViNE* to scale to larger problem instances.

Finally, we perform extensive simulations to evaluate the optimality and scalability of the proposed solutions under single and double substrate link failures. We restrict our simulations to one and two link failure cases since the probability of more than two simultaneous link failures is extremely low [117, 65]. We also compare our solutions with a VNE approach that does not guarantee VN connectivity upon failures [185]. Although our sequential solutions (*i.e.*, *CoViNE-ILP* and *CoViNE-fast*) do not guarantee optimality, simulation results show that they perform close to *CoViNE-opt*, and scale to larger problem instances. Finally, we demonstrate how *CoViNE* can enable a VN operator to recover from link failures without depending on the SN provider.

The rest of this chapter is organized as follows. We discuss the related literature in Section 2.1. In Section 2.2, we present the system model, *CoViNE* problem statement, definitions, and assumptions. An ILP formulation for optimally solving *CoViNE* is presented in Section 2.3. The theoretical analysis of *CoViNE* is laid in Section 2.4. Then, we present a heuristic algorithm for VN augmentation and computing disjointness constraints in Section 2.5.1, followed by an ILP formulation and a heuristic algorithm for *CoViNE* embedding in Section 2.5.2 and in Section 2.5.3, respectively. Evaluation results for *CoViNE* are presented in Section 2.6. Finally, we conclude the chapter in Section 2.7.

2.1 Related Works

Survivability in NV and IP-over-WDM networks has been well studied over the past years [131, 182, 103, 120, 71, 171, 94, 145]. We discuss the most prominent approaches

addressing survivability from both NV (§ 2.1.1) and IP-over-WDM literature (§ 2.1.2), and contrast them with our solutions for *CoViNE*.

2.1.1 Survivable Virtual Network Embedding (SVNE)

Rahman *et al.*, first formulated the SVNE for single substrate link failure as a mixed integer liner program [131]. Subsequent research works have addressed different aspects of SVNE such as substrate node failures [176, 177, 112, 47] and link failures [71, 150, 40, 94]. SVNE approaches for node failures can be broadly classified into two groups. The first group of works proposed to pro-actively provision dedicated or shared resources in the SN to survive single or multiple node failures [176, 177, 47]. The second group proposed to reactively compute VN embedding after one or more nodes have failed in the SN [112, 128, 140]. Another stream of SVNE research has focused on ensuring VN survivability during one or more substrate link failures, both pro-actively during VN embedding [71, 94, 40, 145] and reactively after a failure [80, 150]. In contrast, we address multiple substrate link failures and propose a different form of survivability instead of guaranteeing full bandwidth of the virtual links during failures as considered in the SVNE approaches.

Recently, there has been a growing interest in designing connectivity-guaranteed VNE schemes [183, 184]. For instance, the proposal from Zhu *et al.* [184] ensures that the unaffected part of a VN remains connected under a single substrate node failure to continue the services provided by the VN. They formulate the problem as an ILP and develop an ant colony optimization algorithm to obtain a local optimal solution. Hmaity *et al.* [79] distinguish between network connectivity and content connectivity and focus on providing content connectivity against double link failures. Their approach guarantees connectivity in a VN after single substrate link failure and maintains content connectivity in the presence of double substrate link failures. They argue that guaranteeing content connectivity may require lower amount of resources compared to network connectivity, and can be the sought after choice for Content Delivery Networks. Zhou *et al.* [183] propose to use cross-layer spanning trees to design survivable cloud networks against SRLG failure. Each spanning tree protects one SRLG by mapping the virtual links in the spanning tree in the paths that avoid the substrate links in the SRLG. They also discuss a way to extend their SRLG based solution to k substrate link failures. However, this solution needs a combinatorial number of spanning trees to protect in order to survive arbitrary k link failures. In contrast, our solution computes only one spanning tree of the VN even to survive any k substrate link failures.

2.1.2 Survivability in IP-over-WDM Network

Modiano *et al.* [120] presented an ILP formulation for survivable link routing of an IP network on WDM SN in the presence of a WDM link failure. Their formulation explores exponential number of edge-cuts in the IP network and ensures that the IP links belonging to a edge-cut are routed on at least two disjoint WDM paths. This approach does not scale well as the number of edge-cuts grows exponentially with the size of the IP network. Todimala *et al.* [159] have proposed an improved ILP formulation by identifying polynomial number of primary cuts in planar and hierarchical planar cyclic graphs representing IP networks. Their formulation computes the survivable routing of these sub-classes of IP networks against single node or SRLG failure(s). However, this formulation is not applicable to non-planar IP topologies and arbitrary failure scenarios. Lee *et al.* [107] extended the Max-flow min-cut theorem to multi-layer networks and proposed approximation algorithms for survivable IP network mapping on an WDM SN, while maximizing the minimum cross layer cut. Zhou *et al.* [181] identified four cross-layer metrics and presented mixed-integer linear programs (MILPs) to determine a mapping that maximizes one of the defined metrics. They also provided an MILP formulation for augmenting the IP topology. A major drawback of these MILP-based approaches is that they do not scale well with VN or SN sizes, because of the inherent complexity of the LP-solvers.

Several heuristic based approaches have been proposed for survivable IP link mapping in large WDM networks. Kurant *et al.* [103, 104] proposed SMART, a framework for finding survivable mapping of an IP network by repeatedly picking sub-graph (e.g., cycles) of the network and mapping the IP links in the sub-graph on disjoint paths in the WDM SN. SMART can ensure connectivity under double WDM link failures for IP networks having a few special structures, thus limiting its applicability. An extension to SMART has been proposed in [158] that exploits the duality between circuits and cuts in the graph representing an IP topology. In [157], Thulasiraman *et al.* remove some of the shortcomings of the dual framework of SMART by using generalized circuit and cutset cover sequences. In another extension of SMART, Javed *et al.* [86] used the concept of randomized rounding discussed in [130] to find disjoint WDM paths for the IP links that achieved higher success rate than SMART. Zhou *et al.* [182] proposed an algorithm that identifies a set of spanning trees of an IP network and computes a shortest-path based routing of the IP links such that at least one of the spanning trees remains unaffected after a WDM link failure. Another school of thought is to compute restoration path for each IP link in a way that the IP link and its restoration path are not affetced by the same substrate link failures [109, 85]. However, these approaches require to know the set of substrate links that will potentially fail together. In contrast, our solution is generic, *i.e.*, does not assume any specific property

of the IP network or the WDM SN, and can ensure connectivity in the presence of multiple WDM link failures.

2.1.3 Virtual Link Augmentation

IP link augmentation strategies such as [182, 110, 105] to survive WDM link failures primarily focus on single link failure scenarios and cannot be generalized to multiple failures. Zhou *et al.*, propose to augment logical VLinks between arbitrary pairs of VNodes [158]. In contrast, we propose to perform augmentation only between pairs of adjacent VNodes not to alter the VN topology. Thulasiraman *et al.*, propose an augmentation strategy for ensuring survivability under k WDM link failures [156]. They propose to augment IP links until a complete subgraph of $k+2$ nodes in the IP network is constructed and the remaining nodes are $k+1$ edge connected to the subgraph. Their solution maps all the IP links in the complete subgraph of $k+2$ nodes onto mutually disjoint WDM paths. To survive k link failures, this approach requires higher number of IP links to be augmented and more disjointness constraints to be satisfied than those of our approach.

2.1.4 Complexity of the VNE problem

Optimally solving the general case of the VNE problem without any survivability requirements, is at least as hard as the NP-Hard *Multi-commodity Unsplittable Flow Problem (MCUFP)* [58] when the source and destination of the flows are unknown. The best known approximation bound for the MCUFP with known sources and destinations is (2 + ϵ) for very simple classes of graphs, namely, line and cycle graphs [15]. A Linear Programming relaxation based algorithm for unsplittable flows on trees has been proposed in [61], however, the approximation ratio is a logarithmic function of the number of nodes. In reality, SNs are more densely connected than line and cycle graphs, and trees. Finding a constant factor approximation algorithm for general graphs still remains an open problem [27]. Moreover, line graphs and trees are 1-edge connected, therefore, they are not suitable for deploying SNs to guarantee VN connectivity under $k \geq 1$ link failures. Finally, a more recent study has proved that the general case of the VNE problem with capacity constraints on the links is $\mathcal{NP}$-complete and cannot be approximated under any objective unless $\mathcal{P} = \mathcal{NP}$ [135].

Table 2.1: Notation Table

$G = (V, E)$	Substrate Network (SN)
$\bar{G} = (\bar{V}, \bar{E})$	Virtual Network (VN)
$\hat{G} = (\hat{V}, \hat{E})$	k-protected VN
$\hat{G}_k = (\hat{V}_k, \hat{E}_k)$	k-protected component of a VN $\bar{G}$
$\hat{G}_k \odot \hat{v}$	An expansion of $\hat{G}_k$ towards $\hat{v}$
$\bar{C}_i$	An edge-cut $\bar{C}_i \subset \bar{E}$ s.t. $\lvert C_i \rvert > 0$ in $\bar{G}$
$\bar{C}^{\bar{G}}$	Set of edge-cuts $\bar{C}_i \subset \bar{E}$ s.t. $\lvert C_i \rvert > 0$ in $\bar{G}$
$\bar{C}_i \cup \bar{C}_i^k$	The set of VLinks in $\bar{C}_i \in \bar{C}^{\bar{G}}$ and all the parallel VLinks augmented to the VLinks in $\bar{C}_i \in \bar{C}^{\bar{G}}$
$(\bar{u}, \bar{v})^k$	k-th VLink between $\bar{u}$ and $\bar{v}$
$\chi^{\hat{u}\hat{v}k}$	Conflicting set of a VLink $(\hat{u}, \hat{v})^k$
$\chi_{\odot}^{\hat{u}\hat{v}k}$	Conflicting set of a VLink $(\hat{u}, \hat{v})^k$ during expansion
$\chi^{\hat{G}}$	Conflicting set of a VN $\hat{G}$
Q^{uv}	A path between SNodes u and v in G
$P^{\hat{u}\hat{v}}$	A path between VNodes $\hat{u}$ and $\hat{v}$ in $\hat{G}$
$\mathscr{P}^{\hat{u}\hat{v}}$	Set of edge-disjoint paths between $\hat{u}$ and $\hat{v}$ in $\hat{G}$
$P_i^{\hat{u}\hat{v}}$	i-th edge-disjoint path from $\hat{u}$ to $\hat{v}$ in $\hat{G}$
$\mathtt{p}_i^{\hat{u}\hat{v}}$	i-th edge-disjoint shortest path from $\hat{u}$ to $\hat{v}$ in $\hat{G}$
$\mathcal{P}^{\hat{G}_k\hat{v}}$	Set of edge-disjoint shortest paths from $\hat{v}$ to $\hat{G}_k$
K	Set of integers from 0 to k, *i.e.*, $\mathcal{Z} \cap [0, k]$

2.2 Preliminaries

The subsequent sections build upon the background, definitions, and assumptions presented in this section. Table 2.1 lists all the major notations used in the rest of this chapter.

2.2.1 System Model

We represent an SN as an undirected graph, $G = (V, E)$, where V and E denote the set of Substrate Nodes (SNodes) and Substrate Links (SLinks), respectively. The set of neighbors of an SNode $u \in V$ is denoted by $\mathcal{N}(u)$. Bandwidth capacity of an SLink $(u, v) \in E$ is

b_{uv}, while the cost of allocating one unit of bandwidth in (u, v) is C_{uv}. Similarly, a VN is represented as an undirected graph $\bar{G} = (\bar{V}, \bar{E})$, where $\bar{V}$ and $\bar{E}$ denote the set of Virtual Nodes (VNodes) and Virtual Links (VLinks), respectively. The set of neighbors of a VNode $\bar{v} \in \bar{V}$ is denoted by $\mathcal{N}(\bar{v})$. Each VLink $(\bar{u}, \bar{v}) \in \bar{E}$ has bandwidth requirement $b_{\bar{u}\bar{v}}$. Each VNode $\bar{u} \in \bar{V}$ has a location constraint, $L(\bar{u}) \subseteq V$, that denotes the set of SNodes where $\bar{u}$ can be embedded. We represent the location constraint $L(\bar{u}) \subseteq V$ of $\bar{u} \in \bar{V}$ with the binary variable $\ell_{\bar{u}u}$ that is set to 1 if $\bar{u} \in \bar{V}$ can be mapped to $u \in V$, 0 otherwise. Let Q^{uv} represent a path in the SN between a pair of SNodes $u \in V$ and $v \in V$ such that $u \neq v$..

2.2.2 CoViNE Problem Statement

Given an SN $G = (V, E)$, a VN $\bar{G} = (\bar{V}, \bar{E})$, and location constraints $L(\bar{u})$, $\forall \bar{u} \in \bar{V}$, CoViNE finds an embedding that

- provides a function $f : \bar{V} \rightarrow V$ to map every VNode $\bar{u} \in \bar{V}$ to exactly one SNode $u \in V$ while satisfying the location constraint and incurring no overlap, *i.e.*, $\forall \bar{u}, \bar{v} \in \bar{V} \wedge \bar{u} \neq \bar{v} \implies f(\bar{u}) \neq f(\bar{v})$ and $\forall \bar{u} \in \bar{V} \quad f(\bar{u}) \in L(\bar{u})$,
- provides a function $g : \bar{E} \rightarrow 2^E$ to map each VLink $(\bar{u}, \bar{v}) \in \bar{E}$ to a substrate path $Q^{f(\bar{u})f(\bar{v})}$ with sufficient bandwidth to satisfy the VLink demand $b_{\bar{u}\bar{v}}$,
- ensures the connectivity in $\bar{G}$ in the presence of up to k SLink failures in G,
- minimizes the total cost of embedding in terms of substrate bandwidth consumption as defined by the following

$$\sum_{\forall (\bar{u},\bar{v}) \in \bar{E}} \sum_{\forall (u,v) \in Q^{f(\bar{u})f(\bar{v})}} C_{uv} \times b_{\bar{u}\bar{v}} \tag{2.1}$$

2.2.3 Definitions and Design Choices

Definition 1. ***Edge-cut:*** *An edge-cut $\bar{C}_i \subset \bar{E}$ of a VN $\bar{G}$ is the set of VLinks that connect the VNodes in a non-empty set $\bar{S} \subset \bar{V}$ to the VNodes in $\bar{V} \setminus \bar{S}$ and the removal of the VLinks in $\bar{C}_i$ partitions the VN. Let $\bar{C}^{\bar{G}}$ be the set of all edge-cuts in the VN $\bar{G}$, $\bar{C}^{\bar{G}} = \{\bar{C}_1, \bar{C}_2, \ldots, \bar{C}_m\}$. Since the number of non-empty proper subsets (i.e., $\bar{S} \neq \phi \wedge \bar{S} \neq \bar{V}$) of $\bar{V}$ is $2^{\bar{V}} - 2$, we have $m = 2^{\bar{V}} - 2$. Let $|\bar{C}_i|$ denote the number of VLinks in the edge-cut $\bar{C}_i$.*

A VN embedding remains connected during k SLink failures if the following two necessary conditions are met: i) the VN is $k+1$ edge-connected following the definition of $k+1$ edge-connected graphs, implying that the size of each edge-cut is at least $k+1$, $\forall C_i \in \bar{C}^{\bar{G}}, |\bar{C}_i| >= k+1$, ii) the VLinks in each edge-cut C_i are embedded on at least $k+1$ edge-disjoint paths in the SN. This can be trivially proved since the failure of k SLinks can impact at most k edge-disjoint paths in the SN, leading to at most k VLink failures from an edge-cut in a VN leaving at least one VLink to ensure connectivity. However, if the given VN $\bar{G}$ lacks $k+1$ connectivity, we need to augment $\bar{G}$ with additional VLinks. This augmentation can be done in two ways: i) augment VLinks between arbitrary pair of VNodes, which is a well studied problem [110, 105]; ii) augment parallel VLinks between already adjacent VNodes in $\bar{G}$ [182, 158, 109]. Arbitrary augmentation can ensure $k+1$ edge connectivity by introducing minimal number of VLinks, however, this approach will change the input VN topology. Although parallel VLink augmentation may not yield minimal resource usage, it does not alter the input VN topology. From a VN operator perspective, it is very important to preserve its VN topology. Hence, we opt for the second alternative, *i.e.*, augmenting a VN with only parallel VLinks.

Definition 2. ***k-protected VN:*** *A k-protected VN, $\hat{G} = (\hat{V}, \hat{E})$, is a VN that becomes $k+1$ edge connected after augmenting the minimum number of parallel VLinks to a VN, $\bar{G} = (\bar{V}, \bar{E})$. The k-th parallel VLink between $\bar{u}$ and $\bar{v}$ is denoted by $(\bar{u}, \bar{v})^k$, where $(\bar{u}, \bar{v})^0$ or simply $(\bar{u}, \bar{v})$ represents the input VLink between $\bar{u}$ and $\bar{v}$. Here, $\hat{V} = \bar{V}$ and $\hat{E} = \bar{E} \cup \tilde{E}$, where $\tilde{E} = \{(\bar{u}, \bar{v})^k | (\bar{u}, \bar{v}) \in \bar{E}, k \in K$ and $k \geq 1\}$ is the set of augmented parallel VLinks.*

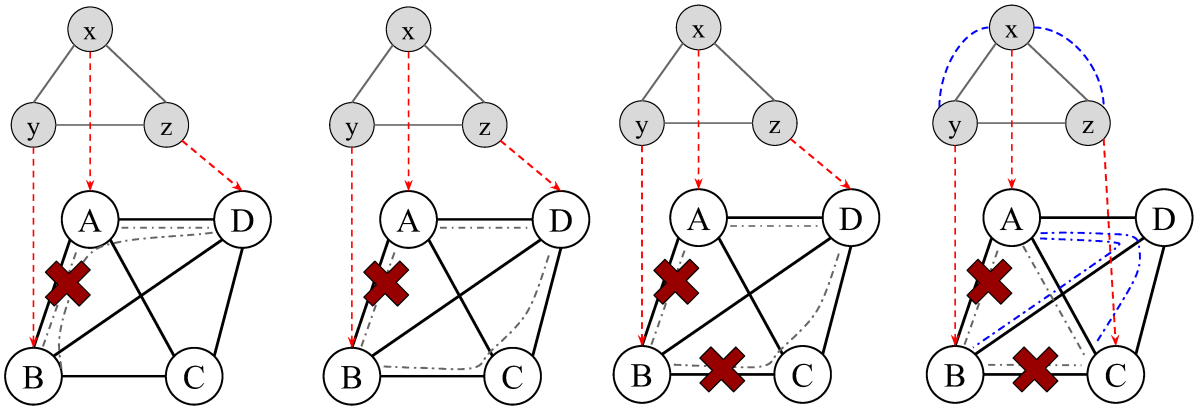
Definition 3. ***k-protected component:*** *A k-protected component of a graph $\bar{G}$ is a multi-graph $\hat{G}_k = (\hat{V}_k, \hat{E}_k)$, where $\hat{V}_k \subseteq \bar{V}$, $\hat{E}_k = \bar{E}_k \cup \tilde{E}_k$, $\bar{E}_k \subseteq \bar{E}$, $\tilde{E}_k \subseteq \tilde{E}$ and $\tilde{E}_k$ is a set of parallel VLinks augmented in such a way that simultaneous removal of k arbitrary VLinks from $\hat{G}_k$ will not partition $\hat{G}_k$.*

Determining the bandwidth of the parallel VLinks as well as the amount of spare bandwidth to be reserved for the input VLinks (in $\bar{E}$) to survive failures is a separate problem of its own, and has been studied in [109, 92, 113, 115]. Here, we assume that the bandwidth of a parallel VLink will be the same as the bandwidth of the input VLink parallel to it in order to salvage full bandwidth of the VLink upon failures. In the case that a parallel VLink has lower bandwidth than the input VLink, the amount of restored bandwidth will be decreased.

2.2.4 CoViNE Example

We illustrate CoViNE examples for single and double failure scenarios in Fig. 2.1(a) and Fig. 2.1(b), respectively. In these examples, xyz is the VN and $ABCD$ is the SN. The arrow from a VNode to an SNode denotes node mapping and the dotted lines between SNodes denote link mapping. To survive single SLink failure ($k = 1$), the VN must be 2 edge-connected. Since xyz is already 2 edge-connected, no augmentation is required. Fig. 2.1(a) shows an un-survivable embedding (on the left) and a survivable embedding (on the right) of xyz. They differ in satisfying disjointness constraints. The embedding on the left satisfies no disjointness constraint, hence the VLinks of an edge-cut $\{(x, y), (y, z)\}$ share an SLink (A, B). Upon the failure of (A, B), both VLinks fail, and VNode y is disconnected from the rest of the VN. The embedding on the right adheres to the disjointness constraints, hence no SLinks are shared. Even though SLink (A, B) and correspondingly VLink (x, y) fail, the VN remains connected.

To survive double link failures (Fig. 2.1(b)), *i.e.*, for $(k = 2)$, the VN should be 3 edge-connected, which is not the case for VN xyz. Due to such lack of edge-connectivity, even an edge-disjoint embedding of all the VLinks (the embedding on the left) cannot survive two SLink failures. Hence, we transform the VN into a 2-protected one by augmenting VLinks (dashed VLinks in the figure) and embedding the resulting VN adhering to the disjointedness constraints as shown on the right. Note that, for a survivable embedding of the 2-protected VN, some SLinks can be shared (*e.g.*, (A, D)) among the mappings of some VLinks since not all the VLinks need to be embedded on mutually disjoint paths.



(a) Single Failure (b) Double Failure

Figure 2.1: CoViNE examples

2.3 Optimal Solution to CoViNE

In this section, we present *CoViNE-opt*, an ILP formulation for optimally solving *CoViNE*. *CoViNE-opt* transforms an input VN $\bar{G}$ to a k-protected VN $\hat{G}$ by augmenting parallel VLinks, and minimizes the total cost of provisioning bandwidth for the VLinks of $\hat{G}$ on an SN G, while ensuring $\hat{G}$'s connectivity in the presence of k SLink failures.

2.3.1 Decision Variables

Recall from Section 2.2.3 that $\bar{G}$ needs to be augmented with additional VLinks if $\bar{G}$ is not already $k+1$ edge-connected. This implies that we must add parallel VLinks to the edge-cuts in $\bar{G}$ whose sizes are less than $k+1$, thus ensuring that there are no edge-cuts in $\bar{G}$ with size less than $k+1$. Determining which parallel VLinks should be added to $\bar{G}$ is non-trivial as augmenting one VLink parallel to $(\bar{u}, \bar{v})$ may increase the sizes of all edge-cuts that contain $(\bar{u}, \bar{v})$, resulting in a combinatorial decision making problem. To optimally decide which parallel VLinks are augmented, we introduce the following decision variable:

$$a_k^{\bar{u}\bar{v}} = \begin{cases} 1 & \text{if } k\text{-th VLink between } \bar{u} \text{ and } \bar{v} \text{ s.t. } (\bar{u}, \bar{v}) \in \bar{E} \\ & \text{is augmented,} \\ 0 & \text{otherwise.} \end{cases}$$

Note that, $a_0^{\bar{u}\bar{v}}$ denotes the input VLink between $\bar{u}$ and $\bar{v}$. Therefore, $\forall (\bar{u}, \bar{v}) \in \bar{E}, a_0^{\bar{u}\bar{v}} = 1$. Once a VLink $(\bar{u}, \bar{v})^k$ s.t. $(\bar{u}, \bar{v}) \in \bar{E} \wedge k \in K \wedge k \neq 0$ is augmented by *CoViNE-opt*, $(\bar{u}, \bar{v})^k$ is included in all the edge-cuts that contain $(\bar{u}, \bar{v})$. We denote the set of VLinks in $\bar{C}_i$ and all the parallel VLinks augmented to the VLinks in $\bar{C}_i$ as $\bar{C}_i \cup \bar{C}_i^k$. Mathematically, $\bar{C}_i \cup \bar{C}_i^k = \{\{(\bar{u}, \bar{v})^0 \text{ s.t. } (\bar{u}, \bar{v}) \in \bar{C}_i\} \cup \{(\bar{u}, \bar{v})^k \text{ s.t. } (\bar{u}, \bar{v}) \in \bar{C}_i \wedge k \in K \wedge k \neq 0 \wedge a_k^{\bar{u}\bar{v}} = 1\}\}$.

A VLink is mapped to a set of SLinks forming a path in the SN. In this ILP formulation, we represent a bidirectional SLink by two unidirectional SLinks in opposite directions. We represent the mapping between $(\bar{u}, \bar{v})^k$, the k-th VLink between adjacent VNodes $\bar{u}$ and $\bar{v}$ and an SLink $(u, v) \in E$ using the following decision variable:

$$x_{uv}^{\bar{u}\bar{v}k} = \begin{cases} 1 & \text{if } (\bar{u}, \bar{v})^k \text{ s.t. } (\bar{u}, \bar{v}) \in \bar{E} \text{ and } k \in K \text{ is} \\ & \text{mapped to } (u, v) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

The following decision variable represents the mapping between VNodes and SNodes:

$$y_{\bar{u}u} = \begin{cases} 1 & \text{if } \bar{u} \in \bar{V} \text{ is mapped to } u \in V, \\ 0 & \text{otherwise.} \end{cases}$$

To guarantee that a VN remains connected (*i.e.*, at least one path exists between any pair of VNodes) during any combination of k SLink failures, one of the necessary conditions discussed in Section § 2.2.3 is that for any edge-cut $\bar{C}_i \in C^{\bar{G}}$, the VLinks in $\bar{C}_i \cup \bar{C}_i^k$ are embedded on at least $k+1$ edge-disjoint paths in the SN. *CoViNE-opt* enforces this condition by partitioning the set of VLinks in each $\bar{C}_i \cup \bar{C}_i^k$ into two mutually exclusive groups. The first group, denoted as $D_{\bar{C}_i}$, will contain the VLinks that cannot share an SLink in their mappings to ensure the existence of at least $k+1$ edge-disjoint paths among the mappings of the VLinks in $\bar{C}_i \cup \bar{C}_i^k$. Hence, the size of $D_{\bar{C}_i}$ must be at least $k+1$. The second group will contain the rest of the VLinks in $\bar{C}_i \cup \bar{C}_i^k$ that are not in $D_{\bar{C}_i}$. The following decision variable decides the assignment of a VLink in $\bar{C}_i \cup \bar{C}_i^k$ to $D_{\bar{C}_i}$:

$$d_{\bar{C}_i}^{\bar{u}\bar{v}k} = \begin{cases} 1 & \text{if } (\bar{u},\bar{v})^k \text{ s.t. } (\bar{u},\bar{v}) \in \bar{C}_i \text{ and } k \in K \\ & \text{belongs to } D_{\bar{C}_i}, \\ 0 & \text{otherwise.} \end{cases}$$

2.3.2 Constraints

Augmentation Constraints

Parallel VLinks are augmented to ensure that there is no edge-cut in the VN with less than $k+1$ VLinks. (2.2) ensures that the size of each edge-cut is at least $k+1$ after augmentation. Note that if *CoViNE-opt* decides to augment a parallel VLink, the augmented VLink must also be mapped to a path in the SN (see § 2.3.2), increasing the cost of embedding. Since the objective of *CoViNE-opt* is a minimization function (see § 2.3.3), *CoViNE-opt* will augment the minimum number of parallel VLinks despite not having a strict equality in (2.2).

$$\forall \bar{C}_i \in \bar{C}^{\bar{G}} \text{ s.t. } |\bar{C}_i| < k+1 : \sum_{\forall (\bar{u},\bar{v}) \in \bar{C}_i} \sum_{k \in K} a_k^{\bar{u}\bar{v}} >= k+1 \tag{2.2}$$

VLink Mapping Constraints

VLinks are mapped to substrate paths following a *Multi-commodity Unsplittable Flow* formulation [119]. Bandwidth conservation is ensured by (2.3) which enforces that an SLink is not assigned VLink demands that exceed the SLink's bandwidth capacity. Then, (2.4) ensures flow conservation by making sure that for each input $(a_0^{\bar{u}\bar{v}}, \forall (\bar{u},\bar{v}) \in \bar{E})$ and augmented $(a_k^{\bar{u}\bar{v}} = 1 \text{ s.t. } (\bar{u},\bar{v}) \in \bar{E} \wedge k \in K \wedge k \neq 0)$ VLink, the in-flow and out-flow of each

SNode is equal except at the SNodes where the endpoints of a VLink are mapped. Since the decision of VLink augmentation is not known in advance, the right hand side of (2.4) is multiplied by $a_k^{\bar{u}\bar{v}}$, yielding a quadratic constraint.

$$\forall (u,v) \in E : \sum_{\forall (\bar{u},\bar{v}) \in \bar{E}} \sum_{\forall k \in K} x_{uv}^{\bar{u}\bar{v}k} \times b_{\bar{u}\bar{v}} \leq b_{uv} \tag{2.3}$$

$$\forall (\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : \\ \sum_{\forall v \in \mathcal{N}(u)} (x_{uv}^{\bar{u}\bar{v}k} - x_{vu}^{\bar{u}\bar{v}k}) = a_k^{\bar{u}\bar{v}} \times (y_{\bar{u}u} - y_{\bar{v}u}) \tag{2.4}$$

We take the following steps to linearize (2.4). First, we introduce two binary variables $w_u^{\bar{u}\bar{v}k}$ and $z_u^{\bar{u}\bar{v}k}$ such that:

$$\forall (\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : w_u^{\bar{u}\bar{v}k} \leq a_k^{\bar{u}\bar{v}} \tag{2.5}$$

$$\forall (\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : w_u^{\bar{u}\bar{v}k} \leq y_{\bar{u}u} \tag{2.6}$$

$$\forall (\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : w_u^{\bar{u}\bar{v}k} \geq a_k^{\bar{u}\bar{v}} + y_{\bar{u}u} - 1 \tag{2.7}$$

$$\forall (\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : z_u^{\bar{u}\bar{v}k} \leq a_k^{\bar{u}\bar{v}} \tag{2.8}$$

$$\forall (\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : z_u^{\bar{u}\bar{v}k} \leq y_{\bar{v}u} \tag{2.9}$$

$$\forall (\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : z_u^{\bar{u}\bar{v}k} \geq a_k^{\bar{u}\bar{v}} + y_{\bar{v}u} - 1 \tag{2.10}$$

Then, we rewrite (2.4) in a linear form using the newly introduced variables as follows:

$$\forall (\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall u \in V : \\ \sum_{\forall v \in \mathcal{N}(u)} (x_{uv}^{\bar{u}\bar{v}k} - x_{vu}^{\bar{u}\bar{v}k}) = w_u^{\bar{u}\bar{v}k} - z_u^{\bar{u}\bar{v}k} \tag{2.11}$$

Disjointedness Constraints

(2.12) ensures that for each edge-cut $\bar{C}_i$, the disjoint group $D_{\bar{C}_i}$ contains at least $k+1$ VLinks from $\bar{C}_i \cup \bar{C}_i^k$. To prevent the inclusion of a non-augmented VLink into $D_{\bar{C}_i}$, we multiply $d_{\bar{C}_i}^{\bar{u}\bar{v}k}$ by $a_k^{\bar{u}\bar{v}}$ in (2.12). An SLink (u,v) can be used at most once in the mappings of the VLinks in $\bar{C}_i \cup \bar{C}_i^k$ that are assigned to $D_{\bar{C}_i}$ to ensure their edge-disjoint embedding. We enforce this by (2.13). Since both (2.12) and (2.13) are quadratic, we linearize them using the same technique discussed for (2.4).

$$\forall \bar{C}_i \in \bar{C}^{\bar{G}} : (\sum_{\forall (\bar{u},\bar{v}) \in \bar{C}_i} \sum_{k \in K} d_{\bar{C}_i}^{\bar{u}\bar{v}k} \times a_k^{\bar{u}\bar{v}}) \geq k+1 \tag{2.12}$$

$$\forall (u,v) \in E, \forall \bar{C}_i \in \bar{C}^{\bar{G}} : \sum_{\forall (\bar{u},\bar{v}) \in \bar{C}_i} \sum_{\forall k \in K} d_{\bar{C}_i}^{\bar{u}\bar{v}k} \times (x_{uv}^{\bar{u}\bar{v}k} + x_{vu}^{\bar{u}\bar{v}k}) \leq 1 \tag{2.13}$$

VNode Mapping Constraints

(2.14) ensures that VNode mapping follows the given location constraint. (2.15) ensures that a VNode is mapped to exactly one SNode. Finally, (2.16) ensures that an SNode does not host more than one VNode from a VN. However, an SNode can host multiple VNodes from different VNs.

$$\forall \bar{u} \in \bar{V}, \forall u \in V : y_{\bar{u}u} \leq \ell_{\bar{u}u} \tag{2.14}$$

$$\forall \bar{u} \in \bar{V} : \sum_{u \in V} y_{\bar{u}u} = 1 \tag{2.15}$$

$$\forall u \in V : \sum_{\bar{u} \in \bar{V}} y_{\bar{u}u} \leq 1 \tag{2.16}$$

2.3.3 Objective Function

The objective of *CoViNE-opt* is to minimize the bandwidth provisioning cost over all the SLinks for embedding all the original and augmented VLinks of a VN, $\bar{G}$, subject to the augmentation constraints (§ 2.3.2), VLink mapping constraints (§ 2.3.2), disjointness constraints (§ 2.3.2), and VNode mapping constraints (§ 2.3.2). Given that C_{uv} is the cost of allocating unit bandwidth on SLink $(u,v) \in E$, we have the following objective function for *CoViNE-opt*:

$$\text{minimize} \left(\sum_{\forall (\bar{u},\bar{v}) \in \bar{E}} \sum_{\forall k \in K} \sum_{\forall (u,v) \in E} x_{uv}^{\bar{u}\bar{v}k} \times C_{uv} \times b_{\bar{u}\bar{v}} \right) \tag{2.17}$$

2.3.4 Complexity Analysis

Rost *et al.*, have shown that the VNE problem with capacity constraints on the links, *i.e.*, solving *CoViNE-opt* without the augmentation constraint (2.2) and disjointness constraints (2.12) - (2.13), is $\mathcal{NP}$-complete and cannot be approximated under any objective

unless $\mathcal{P} = \mathcal{NP}$ [135]. Furthermore, when we include the augmentation and disjointedness constraints, *CoViNE-opt* becomes even more computationally intractable. The reason is that *CoViNE-opt* generates an exponential number of variables for $d_{\bar{C}_i}^{\bar{u}\bar{v}k}$ and an exponential number of constraints for (2.2), (2.12), and (2.13) since the number of edge-cuts in a VN is $O(2^{|\bar{V}|})$. For instance, total number of binary variables for $d_{\bar{C}_i}^{\bar{u}\bar{v}k}$ is $|\bar{E}| \times k \times (2^{|\bar{V}|} - 2)$ and total number of constraints for (2.13) is $|E| \times (2^{|\bar{V}|} - 2)$. In the worst case, all binary vectors have to be enumerated and all the constraints need to be explicitly checked for each of the enumeration, yielding the time complexity of $O((|E| \times (2^{|\bar{V}|} - 2)) \times 2^{(|\bar{E}| \times k \times (2^{|\bar{V}|} - 2))})$. To reduce the size of the problem, we decompose the joint optimization, and separate augmentation and disjointness constraints computation from embedding as presented in the subsequent sections. Note that by decomposing *CoViNE-opt* into sub-problems and solving them sequentially may yield sub-optimal solution.

2.4 Theoretical Analysis for k Link Survivability

In this section, we first devise an efficient mechanism to compute disjointness constraints of a VN assuming that the VN is k-protected (§ 2.4.1). We then extend this mechanism to transform an arbitrary VN to a k-protected VN (§ 2.4.2).

2.4.1 Disjointness Constraints Computation

As discussed in § 2.3.4, there are exponential number of edge-cuts in $\hat{G}$ and combinatorial number of ways to assign VLinks from each edge-cut to its $k+1$ disjoint groups. To reduce the exponential number of constraints, we first define the disjointedness relationship between the VLinks of $\hat{G}$ irrespective of the edge-cuts as follows.

Definition 4. ***Conflicting VLinks:*** *Two VLinks are considered conflicting if they must be embedded on edge-disjoint paths in the SN to ensure the VN remains connected (i.e., at least one path exists between any pair of VNodes) in the presence of k SLink failures.*

Definition 5. ***Conflicting set:*** *A conflicting set of a VLink $(\hat{u}, \hat{v})^k$, denoted by $\chi^{\hat{u}\hat{v}k}$, is the set of VLinks in $\hat{E}$ that are conflicting with $(\hat{u}, \hat{v})^k$. A conflicting set of a VN $\hat{G} = (\hat{V}, \hat{E})$, denoted by $\chi^{\hat{G}}$, is defined as $\chi^{\hat{G}} = \{\chi^{\hat{u}\hat{v}k} | \forall (\hat{u}, \hat{v})^k \in \hat{E}\}$.*

A conflicting set of $\hat{G}$ imposes disjointedness constraints on VLink embedding of $\hat{G}$. The larger the size of conflicting sets of the VLinks in $\hat{G}$, the higher the number of disjointedness

constraints to be satisfied, and hence the longer becomes the substrate paths used for VLink embedding. This increased number of disjointedness constraints can have a twofold impact on embedding. First, it can increase the cost of embedding due to the longer substrate paths. Second, and more importantly, it can lead to infeasible solutions due to the lack of adequate edge-disjoint paths in a moderately dense SN. Therefore, we define the notion of *minimal conflicting set* of $\hat{G}$ that ensures $k+1$ edge connectivity of $\hat{G}$ after embedding, while minimizing the requirement of having disjoint paths in the embedding. This can be obtained by finding the minimum number of partitions of the VLinks of $\hat{E}$ such that the VLinks in a partition are not conflicting with one another. Since VLinks in the same partition do not impose any disjointedness constraint, minimizing the number of partitions will yield a minimal conflicting set. However, partitioning the VLinks to yield a minimal conflicting set is non-trivial as per the following theorem.

Theorem 1. *Computing a minimal conflicting set of a VN is NP-complete.*

Proof. This problem is clearly in NP because we can verify that a given conflicting set of $\hat{G}$ ensures $k+1$ edge connectivity in polynomial time by successively removing a VLink $\hat{e_i} \in \hat{E}$ and all the VLinks in $\chi^{\hat{e_i}}$, and then checking for VN connectivity. To show that the problem is NP-complete, we reduce the NP-complete *Minimum vertex coloring* problem [51] to computing a minimal conflicting set. Consider a graph $H = (V_H, E_H)$ as an instance of the *Minimum vertex coloring* problem. Also consider a bijection $\xi : \hat{E} \rightarrow V_H$ that maps each VLink in $\hat{E}$ to a vertex in V_H. There is an edge $(\xi(\hat{e_i}), \xi(\hat{e_j})) \in E_H$ between the vertices $\xi(\hat{e_i}) \in V_H$ and $\xi(\hat{e_j}) \in V_H$ *if and only if* two VLinks $\hat{e_i}, \hat{e_j} \in \hat{E}$ are conflicting with each other. We can test if two VLinks are conflicting in polynomial time by removing them and checking for $k+1$ edge connectivity in $\hat{G}$. Hence, the conflicting set of $\hat{e_i}$ can be computed from H as $\chi^{\hat{e_i}} = \{\hat{e_j} | (\xi(\hat{e_i}), \xi(\hat{e_j})) \in E_H\}$, while the conflicting set of $\hat{G}$ can be constructed as $\chi^{\hat{G}} = \{\chi^{\hat{e_i}} | \forall \hat{e_i} \in \hat{E}\}$. If two VLinks $\hat{e_i}, \hat{e_j} \in \hat{E}$ are not conflicting with each other, there is no edge between $\xi(\hat{e_i})$ and $\xi(\hat{e_j})$. Hence, $\xi(\hat{e_i})$ and $\xi(\hat{e_j})$ can be given the same color in H. Thus, finding a partition of $\hat{E}$ consisting of non-conflicting VLinks is equivalent to finding vertices with the same color in V_H. Therefore, a minimal conflicting set of $\hat{G}$ yields the minimum vertex coloring of H. Hence, *Minimum vertex coloring* $\leq_P$ computing *minimal conflicting set.* □

Note that computing the optimal conflicting set of a VN is harder than the NP-complete problem of computing a minimal conflicting set, since the former takes SN and embedding cost into account in addition to the number of disjointness constraints. Hence, we propose a heuristic algorithm in § 2.5.1 to compute a conflicting set that tries to minimize number of disjointedness constraints as well as embedding cost. The following Theorem, known as

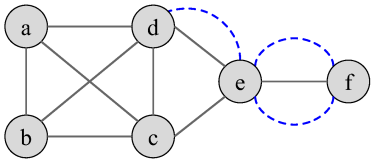


Figure 2.2: The VN with only solid edges is the input VN, $\bar{G}$. The VN with both solid edges ($\bar{E}$) and dashed edges ($\tilde{E}$) is the 2-protected VN, $\hat{G}$. Any subgraph of $\hat{G}$ having 3 edge connectivity is $\hat{G}_2$.

Menger's Theorem [5], provide the basis for our algorithm to compute the conflicting set of a VN $\hat{G}$ within a reasonable time.

Theorem 2. *([5]) The size of the minimum edge-cut for two distinct VNodes $\hat{u}, \hat{v} \in \hat{G}$ is equal to the maximum number of edge-disjoint paths between $\hat{u}$ and $\hat{v}$ in $\hat{G}$.*

According to Theorem 2 any pair of VNodes $\hat{u}$ and $\hat{v}$ in $\hat{G}$ will remain connected in the presence of k SLink failures, if at least one of the edge-disjoint paths $P_i^{\hat{u}\hat{v}} \in \mathscr{P}^{\hat{u}\hat{v}}$ remains intact. This can be achieved by mapping any $k+1$ paths in $\mathscr{P}^{\hat{u}\hat{v}}$ into $k+1$ edge-disjoint paths in the SN. There are a combinatorial number of ways of choosing these $k+1$ edge-disjoint paths between $\hat{u}$ and $\hat{v}$. If $\mathscr{P}_1^{\hat{u}\hat{v}} = P_1^{\hat{u}\hat{v}}, P_2^{\hat{u}\hat{v}}, ..., P_{k+1}^{\hat{u}\hat{v}}$ is one possible combination chosen to have edge-disjoint mapping, two VLinks $(\hat{x},\hat{y})^q \in P_i^{\hat{u}\hat{v}}$ and $(\hat{w},\hat{z})^r \in P_j^{\hat{u}\hat{v}}$, such that $x \neq w$ and $y \neq z$, cannot share an SLink in their mappings. Therefore, a VLink $(\hat{x},\hat{y})^q \in P_i^{\hat{u}\hat{v}}$ is conflicting with all other VLinks present in the paths in $\mathscr{P}_1^{\hat{u}\hat{v}} \setminus P_i^{\hat{u}\hat{v}}$, leading to $|\chi^{\hat{x}\hat{y}q}| = \sum_{P_i^{\hat{u}\hat{v}} \in \mathscr{P}_1^{\hat{u}\hat{v}} \wedge (\hat{x},\hat{y})^q \notin P_i^{\hat{u}\hat{v}}} |P_i^{\hat{u}\hat{v}}|$. For example, in Fig. 2.2, VNodes a and b will remain connected in presence of two SLink failures if the VLinks on paths $P_1^{ab} = (a,b)$, $P_2^{ab} = \{(a,d),(d,c),(c,b)\}$, and $P_3^{ab} = \{(a,c),(c,e),(e,d),(d,b)\}$ are mapped to disjoint SN paths. Hence, $\chi^{ab} = P_2^{ab} \cup P_3^{ab}$.

We now discuss some heuristics to reduce the above computation. First, we can ensure connectivity in $\hat{G}$ by ensuring connectivity in a minimum spanning tree (MST) $\hat{T}$ of $\hat{G}$. In this case, we need to compute $k+1$ edge-disjoint paths only for the $|\hat{V}|-1$ VLinks in $\hat{T}$, as opposed to considering all the VLinks in $\hat{G}$. For instance, in Fig. 2.2, $k+1$ edge-disjoint path computations are required for the VLinks in $\hat{T} = \{(a,b),(a,c),(c,d),(d,e),(e,f)\}$ instead of all the 12 VLinks in $\hat{G}$. Second, instead of arbitrarily selecting $k+1$ edge-disjoint paths from $\mathscr{P}^{\hat{u}\hat{v}}$, we can choose the first $k+1$ edge-disjoint shortest paths between $\hat{u}$ and $\hat{v}$. Thus, the size of the conflicting set of a VLink $(\hat{u},\hat{v})^q \in \hat{T}$ becomes $|\chi^{\hat{u}\hat{v}q}| = \sum_{\mathtt{p}_i^{\hat{u}\hat{v}} \in \mathscr{P}^{\hat{u}\hat{v}} \wedge (\hat{u},\hat{v})^q \notin \mathtt{p}_i^{\hat{u}\hat{v}}}^{i \leq k+1} |\mathtt{p}_i^{\hat{u}\hat{v}}|$, where $\mathtt{p}_i^{\hat{u}\hat{v}}$ is the i-th edge-disjoint shortest path between two adjacent VNodes $\hat{u}$ and $\hat{v}$. This method yields smaller conflicting sets than selecting arbitrary edge-disjoint paths.

For instance, the conflicting set of VLink (a, b) in Fig. 2.2 is $\chi^{ab} = \mathtt{p}_2^{ab} \cup \mathtt{p}_3^{ab}$ where $\mathtt{p}_2^{ab} = \{(a, c), (c, b)\}$, and $\mathtt{p}_3^{ab} = \{(a, d), (d, b)\}$. The following definitions, lemmas, and theorem formalize our heuristics and prove that they result in better conflicting sets than individual computation.

Definition 6. ***Expansion Operator*** $\odot$*: Given a k-protected component* $\hat{G}_k$ *of a VN* $\hat{G}$ *and a VNode* $\hat{v}$*, such that* $\hat{v} \in \hat{V} \setminus \hat{V}_k$ *and* $\exists \hat{u} \in \hat{V}_k, \hat{v} \in \mathcal{N}(\hat{u})$*, we define* $\hat{G}_k \odot \hat{v}$ *as an expansion of* $\hat{G}_k$ *generated by adding* $\hat{v}$ *and all the incident VLinks on* $\hat{v}$ *from any VNode in* $\hat{G}_k$*. Mathematically,* $\hat{G}_k \odot \hat{v} = (\hat{V}_k \cup \{\hat{v}\}, \hat{E}_k \cup \{(\hat{u}, \hat{v})^q | \hat{u} \in \hat{V}_k, \hat{u} \in \mathcal{N}(\hat{v})\})$

Definition 7. ***EDSP*** $\mathcal{P}^{\hat{G}_k\hat{v}}$***:*** *We define EDSP as a set of Edge-Disjoint Shortest Paths* $\mathcal{P}^{\hat{G}_k\hat{v}} = \{\mathtt{p}_i^{\hat{x}\hat{v}}\}$ *between* $\hat{G}_k$ *and a VNode* $\hat{v} \in \hat{V} \setminus \hat{V}_k$*, such that* $\hat{x} \in \hat{V}_k$ *and all* $\mathtt{p}_i^{\hat{x}\hat{v}}$ *terminate as the first VNode* $\hat{x}$ *in* $\hat{V}_k$ *is encountered, i.e., the only VNode from* $\hat{V}_k$ *that is on* $\mathtt{p}_i^{\hat{x}\hat{v}}$ *is* $\hat{x}$*.*

Observation 1: Using the expansion lemma, it can be shown that $\hat{G}_k \odot \hat{v}$ is a k-protected component if and only if there exists $k + 1$ edge-disjoint paths from $\hat{G}_k$ to $\hat{v}$ in $\hat{G}$.

Lemma 1. *In an expansion* $\hat{G}_k \odot \hat{v}$*, a VLink in* $(\hat{x}, \hat{y})^q \in \hat{E}_k$ *can not be present in one of the* $k + 1$ *EDSPs in* $\mathcal{P}^{\hat{G}_k\hat{v}}$*.*

Proof. This proof is based on the observation that $\hat{v}$ may not have shortest paths to some of the VNodes in $\hat{G}_k$. We consider an arbitrary VLink $(\hat{x}, \hat{y})^q \in \hat{E}_k$. There can be three possibilities: i) there are two edge-disjoint paths from $\hat{x}$ and $\hat{y}$ to $\hat{v}$, $P^{\hat{x}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}}$ and $P^{\hat{y}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}}$, respectively, such that $(\hat{x}, \hat{y})^q \notin P^{\hat{x}\hat{v}} \wedge (\hat{x}, \hat{y})^q \notin P^{\hat{y}\hat{v}}$. To ensure edge disjointness, $(\hat{x}, \hat{y})^q$ can only be added to any of $P^{\hat{x}\hat{v}}$ or $P^{\hat{y}\hat{v}}$. Adding $(\hat{x}, \hat{y})^q$ to one of $P^{\hat{x}\hat{v}}$ or $P^{\hat{y}\hat{v}}$ increases the length of respective path. ii) There is only one path from $\hat{x}$ (or $\hat{y}$) to $\hat{v}$, $P^{\hat{x}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}}$ (or $P^{\hat{y}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}}$), such that $(\hat{x}, \hat{y})^q \notin P^{\hat{x}\hat{v}}$ (or $(\hat{x}, \hat{y})^q \notin P^{\hat{y}\hat{v}}$) and there is no path from $\hat{y}$ (or $\hat{x}$) to $\hat{v}$ excluding $(\hat{x}, \hat{y})^q$. Again, adding $(\hat{x}, \hat{y})^q$ to $P^{\hat{x}\hat{v}}$ (or $P^{\hat{y}\hat{v}}$) does not contribute in finding a new edge-disjoint path and only increases the length of the path $P^{\hat{x}\hat{v}}$ (or $P^{\hat{y}\hat{v}}$). iii) There is no edge-disjoint path from $\hat{x}$ or $\hat{y}$ to $\hat{v}$. In this case, $(\hat{x}, \hat{y})^q$ can not be present in any of the $k + 1$ EDSPs in $\mathcal{P}^{\hat{G}_k\hat{v}}$. □

Lemma 2. *In an expansion* $\hat{G}_k \odot \hat{v}$*, the size of the conflicting set of a VLink* $(\hat{u}, \hat{v})^q \in \hat{E} \setminus \hat{E}_k$ *is given by*
$|\chi_{\odot}^{\hat{u}\hat{v}q}| = \sum_{\mathtt{p}_i^{\hat{u}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}} \wedge (\hat{u},\hat{v})^q \notin \mathtt{p}_i^{\hat{u}\hat{v}}}^{i \leq k+1} |\mathtt{p}_i^{\hat{u}\hat{v}}|$*, where* $\hat{u} \in \hat{V}_k$ *and* $\hat{v} \in \mathcal{N}(\hat{u})$*.*

Proof. For the embedding of $\hat{G}_k \odot \hat{v}$ on G to remain connected in the presence of k SLink failures, we need to satisfy two conditions: i) at least $k+1$ edge-disjoint paths from $\hat{v}$ to $\hat{G}_k$

exist (*i.e.*, $|\mathcal{P}^{\hat{G}_k\hat{v}}| \geq k+1$), and ii) all of these paths are embedded on $k+1$ edge-disjoint paths in G. Therefore, a VLink $(\hat{u},\hat{v})^q \in \mathtt{p}_i^{\hat{u}\hat{v}}$ is conflicting with all the VLinks present in the $k+1$ EDSPs in $\mathcal{P}^{\hat{G}_k\hat{v}} \setminus \mathtt{p}_i^{\hat{u}\hat{v}}$. This leads to $|\chi_{\odot}^{\hat{u}\hat{v}q}| = \sum_{\mathtt{p}_i^{\hat{u}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}} \wedge (\hat{u},\hat{v})^q \notin \mathtt{p}_i^{\hat{u}\hat{v}}}^{i \leq k+1} |\mathtt{p}_i^{\hat{u}\hat{v}}|$. □

Theorem 3. *For any VLink $(\hat{u},\hat{v})^q$, the size of a conflicting set $\chi_{\odot}^{\hat{u}\hat{v}q}$ obtained through the expansion of $\hat{G}_k \odot \hat{v}$ is less than or equal to the size of any conflicting set $\chi_I^{\hat{u}\hat{v}q}$ of the same VLink when computed independently, i.e., $|\chi_E^{\hat{u}\hat{v}q}| \leq |\chi_I^{\hat{u}\hat{v}q}|$.*

Proof. We consider two VNodes $\hat{u} \in \hat{V}_k$ and $\hat{v} \in \hat{V} \setminus \hat{V}_k$, such that $\hat{v} \in \mathcal{N}(\hat{u})$. When computed independently, the size of the conflicting set of $(\hat{u},\hat{v})^q$ is $|\chi_I^{\hat{u}\hat{v}q}| = \sum_{\mathtt{p}_i^{\hat{u}\hat{v}} \in \mathscr{P}^{\hat{u}\hat{v}} \wedge (\hat{u},\hat{v})^q \notin \mathtt{p}_i^{\hat{u}\hat{v}}}^{i \leq k+1} |\mathtt{p}_i^{\hat{u}\hat{v}}|$. On other hand, when we construct conflicting set through the expansion, $\hat{G}_k \odot \hat{v}$, the size of the conflicting set of the VLink $(\hat{u},\hat{v})^q \in \hat{E} \setminus \hat{E}_k$ is $|\chi_{\odot}^{\hat{u}\hat{v}q}| = \sum_{\mathtt{p}_i^{\hat{u}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}} \wedge (\hat{u},\hat{v})^q \notin \mathtt{p}_i^{\hat{u}\hat{v}}}^{i \leq k+1} |\mathtt{p}_i^{\hat{u}\hat{v}}|$ (as proven in Lemma 2). In the beginning, when $\hat{G}_k$ contains only one VNode, *i.e.*, $|\hat{V}_k| = 1$, it is obvious that $|\chi_I^{\hat{u}\hat{v}q}| = |\chi_{\odot}^{\hat{u}\hat{v}q}|$. For $|\hat{V}_k| > 1$, consider $\hat{x} \in \hat{V}_k$ such that $\exists \mathtt{p}_i^{\hat{u}\hat{v}} \in \mathscr{P}^{\hat{u}\hat{v}}$ contains $\hat{x}$ and $\mathtt{p}_j^{\hat{x}\hat{v}} \in \mathcal{P}^{\hat{G}_k\hat{v}}$. Since $\mathtt{p}_i^{\hat{u}\hat{v}}$ contains $\hat{x}$, according to the optimal substructure property of the shortest path, we get $\mathtt{p}_i^{\hat{u}\hat{v}} = \mathtt{p}_i^{\hat{u}\hat{x}} || \mathtt{p}_j^{\hat{x}\hat{v}}$, assuming $||$ is the path concatenation operator. Thus, $|\mathtt{p}_j^{\hat{x}\hat{v}}| < |\mathtt{p}_i^{\hat{u}\hat{v}}|$ resulting into $|\chi_{\odot}^{\hat{u}\hat{v}q}| < |\chi_I^{\hat{u}\hat{v}q}|$. If no such $\hat{x}$ is found, we can assume $\hat{x} = \hat{u}$ and in that case $\mathtt{p}_j^{\hat{x}\hat{v}} = \mathtt{p}_i^{\hat{u}\hat{v}}$ yielding $|\chi_{\odot}^{\hat{u}\hat{v}q}| = |\chi_I^{\hat{u}\hat{v}q}|$. Hence, $|\chi_{\odot}^{\hat{u}\hat{v}q}| \leq |\chi_I^{\hat{u}\hat{v}q}|$. □

As an example of Theorem 3, let us consider the VLink (d,e) in Fig. 2.2 and the VN needs to survive single SLink failure. If we compute independently, we get $\chi^{de} = \{(d,c),(c,e)\}$. When we compute through the expansion $\hat{G}_1 \odot e$ where $\hat{V}_1 = \{a,b,c,d\}$, we get $\chi^{de} = \{(c,e)\}$.

2.4.2 VLink Augmentation

As described in § 2.2.3, we may need to augment a given VN $\bar{G}$ with parallel VLinks to transform $\bar{G}$ to a k-protected VN $\hat{G}$. Since augmented parallel VLinks increase both the number of disjointness constraints and embedding cost, it is intuitive to minimize the number of parallel VLinks. Again, we use Theorem 2 to find the pair of VNodes with less than $k+1$ edge connectivity and add parallel VLinks as needed. Assume that for each pair of adjacent VNodes $\bar{u},\bar{v} \in \bar{V}$, there are at least m edge-disjoint paths in $\bar{G}$. If $m \geq k+1$, $\bar{G}$ is at least $k+1$ edge-connected, hence no augmentation is needed. If $m < k+1$, we need to add $k+1-m$ parallel VLinks between $\bar{u}$ and $\bar{v}$. In general, $max(0, k+1-m)$ VLinks are needed for each pair of adjacent VNodes. For instance, a VN should be 3-edge connected to

survive 2 SLink failures. Since there are 2 edge-disjoint paths between d and e in Fig. 2.2, we add a parallel VLink. Similarly, we add 2 parallel VLinks between e and f to make the VN 3-edge connected. No augmentation is required for the rest of the adjacent pair of VNodes. It can be easily shown that the number of VLinks to be augmented remains the same during the expansion, $\hat{G}_k \odot \bar{v}$. In other words, if there are $\hat{m}$ edge-disjoint paths from $\hat{G}_k$ to $\bar{v}$ in $\bar{G}$, augmentation of $max(0, k+1-\hat{m})$ parallel VLinks is needed to ensure the $k+1$ edge connectivity between $\hat{u}$ and $\bar{v}$, where $\hat{u} \in \hat{V}_k$ and $\bar{v} \in \mathcal{N}(\hat{u})$.

Theorem 4. *Given a k-protected component $\hat{G}_k$ and two arbitrary VNodes in $\bar{G}$ such that $\bar{v}_1 \notin \hat{V}_k$ and $\bar{v}_2 \notin \hat{V}_k$; $\hat{G}_k \odot \bar{v}_1 \odot \bar{v}_2$ and $\hat{G}_k \odot \bar{v}_2 \odot \bar{v}_1$ are the resulting k-protected components obtained by incrementally applying expansion operator $\odot$ while considering $\bar{v}_1$ and $\bar{v}_2$ as the initial nodes, respectively; A and B are the ordered set of parallel links (ordered in the sequence they were added) augmented to $\bar{G}$ in the process of obtaining $\hat{G}_k \odot \bar{v}_1 \odot \bar{v}_2$ and $\hat{G}_k \odot \bar{v}_2 \odot \bar{v}_1$, respectively: $|A| = |B|$.*

Proof. To prove the theorem, we assume without loss of generality that an initial k-protected component $\hat{G}_k$ of $\bar{G}$ consists of an arbitrarily selected VNode $\hat{u} \in \bar{V}$ with no VLinks. Let's consider two arbitrary VNodes $\bar{v}_1 \in \mathcal{N}(\hat{u})$ and $\bar{v}_2 \in \mathcal{N}(\hat{u})$. The sets of edge-disjoint paths from $\hat{u}$ to $\bar{v}_1$ and $\bar{v}_2$ are $\mathscr{P}^{\hat{u}\bar{v}_1}$ and $\mathscr{P}^{\hat{u}\bar{v}_2}$, respectively. However, a path $P^{\hat{u}\bar{v}_1} \in \mathscr{P}^{\hat{u}\bar{v}_1}$ does not need to be edge-disjoint with a path $P^{\hat{u}\bar{v}_2} \in \mathscr{P}^{\hat{u}\bar{v}_2}$. Suppose, $|\mathscr{P}^{\hat{u}\bar{v}_1}| = m_1$ and $|\mathscr{P}^{\hat{u}\bar{v}_2}| = m_2$. If either or both of m_1 and m_2 are greater than or equal to $k+1$ then the theorem is trivially proved. Hence we assume that $m_1 < k+1$ and $m_2 < k+1$. We need to show that the order of including $\bar{v}_1$ and $\bar{v}_2$ to $\hat{G}_k$ has no effect on the total number of parallel VLinks needed to get either $\hat{G}_k \odot \bar{v}_1 \odot \bar{v}_2$ or $\hat{G}_k \odot \bar{v}_2 \odot \bar{v}_1$.

If we expand to $\bar{v}_1$ first, $k+1-m_1$ parallel VLinks will be augmented between $\hat{u}$ and $\bar{v}_1$ to get $\hat{G}_k \odot \bar{v}_1$. The parallel VLinks between $\hat{u}$ and $\bar{v}_1$ only contribute in increasing the connectivity between $\hat{u}$ and $\bar{v}_1$, and become part of $\hat{G}_k \odot \bar{v}_1$. Now, we consider $\bar{v}_2$. There can be two possibilities based on whether $\bar{v}_1$ is on a path $P^{\hat{u}\bar{v}_2}$ or not. If $\bar{v}_1$ is on a path $P^{\hat{u}\bar{v}_2}$, $P^{\hat{u}\bar{v}_2} = \{(\bar{u}, \bar{v}_1)\}||P^{\bar{v}_1\bar{v}_2}$. According to Lemma 1, VLinks between $\hat{u}$ and $\bar{v}_1$ are in the k-protected component $\hat{G}_k \odot \bar{v}_1$, and cannot be present in any of the m_2 edge-disjoint paths from $\hat{G}_k \odot \bar{v}_1$ to $\bar{v}_2$, thus the number of edge-disjoint paths between $\bar{v}_2$ and $\hat{G}_k \odot \bar{v}_1$ will remain the same as that of between $\bar{v}_2$ and $\hat{G}_k$. For the second possibility, when $\bar{v}_1$ is not present on any $P^{\hat{u}\bar{v}_2} \in \mathscr{P}^{\hat{u}\bar{v}_2}$, the paths in $\mathscr{P}^{\hat{u}\bar{v}_2}$ will remain unaffected by expansion of $\hat{G}_k \odot \bar{v}_1$. In both cases, the number of edge-disjoint paths from $\hat{G}_k$ to $\bar{v}_2$ remains the same. In other words, the number of parallel VLinks needed to produce $\hat{G}_k \odot \bar{v}_2$ from $\hat{G}_k$ is the same as that for obtaining $\hat{G}_k \odot \bar{v}_1 \odot \bar{v}_2$ from $\hat{G}_k \odot \bar{v}_1$. The same can be shown if we consider $\bar{v}_2$ before $\bar{v}_1$. Therefore, the order of choosing VNodes for inclusion into $\hat{G}_k$

does not have any impact on the number of parallel VLinks to be augmented needed to get either $\hat{G}_k \odot \bar{v}_1 \odot \bar{v}_2$ or $\hat{G}_k \odot \bar{v}_2 \odot \bar{v}_1$. □

2.4.3 Necessary Conditions for a Feasible VN Embedding

VN augmentation satisfies only one of the two necessary conditions for *CoViNE* that is a VN must be $k+1$ edge connected. As we discussed earlier, the other necessary condition for *CoViNE* is to have at least $k+1$ edge-disjoint paths between every pair of VNodes after embedding the VN on the SN. Indeed, an SN that is $k+1$ edge-connected, *i.e.*, each SNode degree is at least $k+1$, satisfies this necessary condition to support the successful embedding of an augmented VN. However, an SN without $k+1$ edge-connectivity between all SNode pairs can also support the successful embedding of an augmented VN as long as there exists at least a node mapping function $f : \bar{V} \rightarrow V$ for which each pair of SNodes $f(\bar{u})$ and $f(\bar{v})$, where $\bar{u}, \bar{v} \in \bar{V} \wedge \bar{u} \neq \bar{v}$, have at least $k+1$ edge-disjoint paths in the SN with sufficient bandwidth. In other words, the SN G must have a non-empty sub-graph with at least $|\bar{V}|$ SNodes that contains at least one SNode from the location constraint sets of each VNode $\bar{u} \in \bar{V}$ and that sub-graph is $k+1$ edge-connected. A brute-force algorithm can enumerate all such sub-graphs of G to compute the optimal solution for *CoViNE*. However, as the number of sub-graphs grows exponentially with the number of nodes in either VN or SN, such brute-force approach cannot scale. In the following, we briefly describe the factors that influence the steps that should be taken to satisfy the aforementioned conditions, which we also exploit to design a scalable heuristic.

Since parallel VLinks added to a VLink $(\hat{u}, \hat{v})$ as part of the graph augmentation provide the required edge-connectivity between the two VNodes $\hat{u}$ and $\hat{v}$, VLink $(\hat{u}, \hat{v})$ and all the parallel VLinks added to $(\hat{u}, \hat{v})$ must be embedded on mutually edge-disjoint paths in the SN. In the worst case, a VLink $(\hat{u}, \hat{v})$ can be augmented with k parallel VLinks requiring $k+1$ edge-disjoint paths between the two SNodes where $\hat{u}$ and $\hat{v}$ are mapped. In addition, conflicting sets may enforce the SPaths used for embedding the augmented VLinks to be disjoint with the embedding of other VLinks of the same VN. Therefore, some of the SPaths used for embedding augmented VLinks can have a large number of SLinks to ensure disjointness with other SPaths, thus resulting in a higher embedding cost. Although the augmentation process described in § 2.4.2 ensures that the number of augmented VLinks remains constant irrespective of initial VNode choice and the subsequent order, the decision of which VLinks to augment is still a combinatorial optimization problem as discussed in § 2.3. This decision of which VLink to augment can have an impact on the subsequent VN embedding since one combination of augmentation may lead to an infeasible embedding due to the lack of sufficient edge-disjoint paths in the SN, while another combination may

result in a higher embedding cost due to using longer edge-disjoint paths. To address these issues, we develop a heuristic algorithm in § 2.5.1 that takes both the existence of sufficient number of edge-disjoint paths in the SN and the number of SLinks present in those edge-disjoint paths into account while expanding a k-protected component $\hat{G}_k$. The heuristic algorithm in § 2.5.1 also computes conflicting sets for each VLink by expanding a k-protected component $\hat{G}_k$ as discussed in § 2.4.1. These conflicting sets can then be used by any embedding algorithm to ensure the existence of $k+1$ edge-disjoint paths between every pair of VNodes in the VN embedding. We also present an ILP formulation and a heuristic algorithm in § 2.5.2 and § 2.5.3, respectively, that leverage conflicting sets of VLinks to compute embedding while satisfying the conditions discussed in this section.

2.5 Sequential Solutions to CoViNE

Due to the intractability of *CoViNE-opt*, in this section, we present two sequential solutions to *CoViNE*. The first solution consists of a heuristic algorithm (Alg. 1) and a simplified formulation of *CoViNE-opt*, namely *CoViNE-ILP*. In this solution, we delegate the task of transforming a VN $\bar{G}$ to a k protected VN $\hat{G}$ and computing conflicting set $\chi^{\hat{G}}$ of $\hat{G}$ to Alg. 1 as described in § 2.5.1. The outputs ($\hat{G}$ and $\chi^{\hat{G}}$) of Alg. 1 are then fed into CoViNE-ILP that embeds a VN on an SN using a multi-commodity flow formulation as presented in § 2.5.2. However, *CoViNE-ILP* cannot scale to larger VNs and SNs due to the limitations of LP solvers. Hence, we propose a heuristic algorithm (Alg. 2) in § 2.5.3 that takes $\hat{G}$ and $\chi^{\hat{G}}$ as inputs and embeds VNodes and disjointness constrained VLinks of $\hat{G}$ in a coordinated manner. Combining Alg. 1 and Alg. 2, we get our last solution, *CoViNE-fast*, that can solve larger problem instances within a reasonable time.

2.5.1 Heuristic Algorithm for Conflicting Set and Augmentation

Alg. 1 starts with a seed k-protected component, $\hat{G}_k$, containing an arbitrary VNode $\bar{u} \in \bar{V}$ with degree at least $k+1$. Then the algorithm adds all of $\bar{u}$'s neighbors $\bar{v} \in \mathcal{N}(\bar{u})$ to $\hat{V}_k$ in the increasing order of a score computed for each VLink $(\bar{u}, \bar{v})$. The score function assigns each VLink $(\bar{u}, \bar{v})$ a value proportional to an estimated cost of embedding $(\bar{u}, \bar{v})$ along with all its augmented VLinks (if necessary) on the SN. Since $\bar{u}$ already belongs to a k-protected component $\hat{G}_k$, the number of additional VLinks that need to be augmented for a VLink $(\bar{u}, \bar{v})$ *s.t.*, $\bar{v} \in \mathcal{N}(\bar{u})$, depends on the degree of $\bar{v}$ denoted by $degree(\bar{v})$. In the worst case, the VLink $(\bar{u}, \bar{v})$ needs to be augmented with $max((k+1-degree(\bar{v})), 0)$ parallel VLinks as $degree(\bar{v})$ imposes an upper bound on the number of EDSPs between $\bar{u}$ and $\bar{v}$. When

Algorithm 1: Compute Conflicting Sets and Augmentation

function *ConflictingSetsAugmentation*($\bar{G}, k$)
  foreach $(\bar{u}, \bar{v}) \in \bar{E}$ **do**
    $\chi^{\bar{u}\bar{v}} \leftarrow \phi$. $a_0^{\bar{u}\bar{v}} \leftarrow 1$, $Q \leftarrow \phi$
  // $\bar{v}$ is an arbitrary VNode with degree$\geq k+1$
  $\exists \bar{v} \in \bar{V} : \hat{G}_k \leftarrow (\{\bar{v}\}, \phi)$
  ENQUEUE(Q, $\bar{v}$)
  while Q *is not empty* **do**
    $\bar{u} \leftarrow DEQUEUE(Q)$
    $\Sigma \leftarrow \{(\bar{u}, \bar{v}) | \bar{v} \in \mathcal{N}(\bar{u}) \wedge \bar{v} \notin \hat{V}_k\}$
    foreach $(\bar{u}, \bar{v}) \in \Sigma$ **do**
      if $degree(\bar{v}) \geq k+1$ **then** $score(\bar{u}, \bar{v}) \leftarrow \infty$
      else
        foreach $(l, m) \in L(\hat{u}) \times L(\hat{v}) \wedge l \neq m$ **do**
          $\mathcal{P}^{lm} \leftarrow$ EDSP$(G, l, m, k+2-degree(\bar{v}))$ **if** $|\mathcal{P}^{lm}| < k+2-degree(\bar{v})$ **then**
            $rank(l, m) \leftarrow \infty$
          else
            $rank(l, m) \leftarrow \sum_{p \in \mathcal{P}^{lm}} |p|$
        $score(\bar{u}, \bar{v}) \leftarrow \min_{(l,m) \in L(\hat{u}) \times L(\hat{v})} rank(l, m) b_{\bar{u}\bar{v}}$
    $\bar{\mathcal{E}} \leftarrow$ Sort Σ in increasing order of $score(\bar{u}, \bar{v})$
    foreach $(\bar{u}, \bar{v}) \in \bar{\mathcal{E}}$ **do**
      $\mathcal{P}^{\hat{G}_k \bar{v}} \leftarrow$ EDSP$(\bar{G}, \hat{G}_k, \bar{u}, \bar{v}, k+1)$
      foreach $i = 1 \rightarrow (k+1-|\mathcal{P}^{\hat{G}_k \bar{v}}|)$ **do**
        $\hat{E} \leftarrow \bar{E} \cup (\bar{u}, \bar{v})^i$, $a_k^{\bar{u}\bar{v}} \leftarrow 1$
        $\mathcal{P}^{\hat{G}_k \bar{v}} \leftarrow \mathcal{P}^{\hat{G}_k \bar{v}} \cup (\bar{u}, \bar{v})^i$
      foreach $\mathtt{p}_i^{\hat{G}_k \bar{v}} \in \mathcal{P}^{\hat{G}_k \bar{v}}$ **do**
        foreach $(\bar{x}, \bar{y})^q \in \mathtt{p}_i^{\hat{G}_k \bar{v}}$ **do**
          foreach $\mathtt{p}_j^{\hat{G}_k \bar{v}} \in \mathcal{P}^{\hat{G}_k \bar{v}} | j \neq i$ **do**
            $\chi^{\bar{x}\bar{y}q} \leftarrow \chi^{\bar{x}\bar{y}} \cup \{\forall (\bar{s}, \bar{t})^r \in \mathtt{p}_j^{\hat{G}_k \bar{v}}\}$
      $\hat{G}_k \leftarrow \hat{G}_k \odot \bar{v}$, ENQUEUE(Q, $\bar{v}$)
  return $\chi^{\bar{G}}$

$degree(\bar{v}) \geq k+1$, no augmentation is needed for the VLink $(\bar{u}, \bar{v})$ and the score function assigns a large value for the estimated embedding cost of $(\bar{u}, \bar{v})$ to make it appear at the end of the sorted order (Line 10). Otherwise, when $degree(\bar{v}) < k+1$, the VLink $(\bar{u}, \bar{v})$ and all the $k+1-degree(\bar{v})$ parallel VLinks need to be embedded on mutually disjoint SPaths following the discussion in § 2.4.3. Therefore, Alg. 1 computes the estimated embedding cost for $(\bar{u}, \bar{v})$ by multiplying the bandwidth demand $b_{\bar{u}\bar{v}}$ with the number of SLinks present in the $k+2-degree(\bar{v})$ EDSPs in the SN that yields the lowest cost embedding for $(\bar{u}, \bar{v})$ and its up to $k+1-degree(\bar{v})$ augmented VLinks (Line 18).

To estimate the lowest embedding cost for $(\bar{u}, \bar{v})$ and its up to $k+1-degree(\bar{v})$ augmented parallel VLinks, Alg. 1 computes the sets of $k+2-degree(\bar{v})$ EDSPs in the SN using `EDSP` procedure for each possible pair of SNodes where VNodes $\bar{u}$ and $\bar{v}$ can be mapped. For a pair of such SNodes l and m, `EDSP` iteratively applies Dijkstra's shortest path algorithm [53] to compute a set of $k+2-degree(\bar{v})$ EDSPs as $\mathcal{P}^{lm}$ (Line 13). After computing each EDSP, all the SLinks present in the EDSP are removed from G in order to ensure the edge-disjointness of the subsequent paths. Alg. 1 then uses the set of $k+2-degree(\bar{v})$ EDSPs with the least number of SLinks over all pairs of $(l, m) \in L(\hat{u}) \times L(\hat{v}) \wedge l \neq m$ to compute the estimated embedding cost of $(\bar{u}, \bar{v})$. Such sorted order of VLinks ensures that parallel VLinks are augmented to VLinks of an MST $\hat{T}$ of $\hat{G}_k$ to ensure necessary connectivity with the lowest estimated cost whenever possible. Such SN awareness also rules out the possibility of augmenting a VLink $(\bar{u}, \bar{v})$ with $k+1-degree(\bar{v})$ parallel VLinks if none of the SNode pairs $(l, m) \in L(\hat{u}) \times L(\hat{v}) \wedge l \neq m$ has at least $k+2-degree(\bar{v})$ EDSPs (Line 15). The final node and link embedding are computed by Alg. 2 that neither uses the estimated embedding cost nor the mappings computed by Alg. 1.

The above process is repeated until all the VNodes of $\bar{G}$ are added to $\hat{G}_k$. For each $\bar{v}$, Alg. 1 computes $k+1$ EDSPs in the VN, $\mathcal{P}^{\hat{G}_k\bar{v}}$ between $\hat{G}_k$ and $\bar{v}$ using `EDSP` procedure (Line 21). `EDSP` initially selects the VLink, $(\bar{u}, \bar{v})$ as the first shortest path $\mathtt{p}_1^{\hat{G}_k\bar{v}}$ to $\mathcal{P}^{\hat{G}_k\bar{v}}$. It then invokes *Dijkstra's shortest path* algorithm [53] k times to compute $\mathtt{p}_i^{\hat{G}_k\bar{v}}$, the i-th EDSP between $\hat{G}_k$ and $\bar{v}$. After computing each $\mathtt{p}_i^{\hat{G}_k\bar{v}}$, all VLinks present in $\mathtt{p}_i^{\hat{G}_k\bar{v}}$ are removed from $\bar{G}$ in order to ensure the edge-disjointness of the later paths. *Dijkstra's shortest path* algorithm is modified to use the bandwidth demand of a VLink as the VLink weight while computing EDSPs. Alg. 1 then proceeds to check if any augmentation is required between $\hat{G}_k$ and $\bar{v}$. To do so, it counts the number of EDSPs, $|\mathcal{P}^{\hat{G}_k\bar{v}}|$, computed in line 21. If the number of EDSPs is less than $k+1$, line 23 of Alg. 1 adds $k+1-|\mathcal{P}^{\hat{G}_k\bar{v}}|$ parallel VLinks between $\bar{u}$ and $\bar{v}$. The i-th parallel VLink, denoted by $(\bar{u}, \bar{v})^i$, constitutes the $(|\mathcal{P}^{\hat{G}_k\bar{v}}|+i)$-th EDSP between $\hat{G}_k$ and $\bar{v}$. Finally, Alg. 1 updates the conflicting sets of the corresponding VLinks as described in Lemma 2 (Line 28).

Discussion

When augmentation is needed, Alg. 1 invokes `EDSP` procedure $O(|\bar{V}||\mathcal{N}(\bar{u})||L(\bar{u})|^2)$ times. `EDSP` invokes *Dijkstra's shortest path* algorithm on SN G $k+1$ times in the worst case. The time complexity of *Dijkstra's shortest path* algorithm on SN G based on a min-priority queue is $O(|E|+|V|\log|V|)$. Therefore, calls to `EDSP` requires running time of $O((k+1)|\bar{V}||\mathcal{N}(\bar{u})||L(\bar{u})|^2(|E|+|V|\log|V|))$. When augmentation is not required, the time complexity of Alg. 1 is dominated by the `EDSP` procedure that is invoked $O(|\bar{V}||\mathcal{N}(\bar{u})|)$ times. `EDSP` invokes *Dijkstra's shortest path* algorithm on VN $\bar{G}$ $k+1$ times yielding $O((k+1)(|\bar{E}|+|\bar{V}|\log|\bar{V}|))$ running time. In this case, the running time of Alg. 1 becomes $O((k+1)|\bar{V}||\mathcal{N}(\bar{u})|(|\bar{E}|+|\bar{V}|\log|\bar{V}|))$.

An alternate way for implementing `EDSP` is to use a maximum flow algorithm [55]. However, the iterative shortest path based method and the maximum flow algorithm have similar running time and solution quality [55]. The only scenario where maximum flow algorithms are known to be more effective is when the graph has a *trap* topology [55]. Dunn *et al.*, have shown that the formation of trap topologies require many conditions to be met simultaneously, hence, they are very rare in production networks [55]. Therefore, taking any special measures for such topology seems unnecessary. Consequently, we made a choice and resorted to using the iterative application of Dijkstra's shortest path algorithm for implementing `EDSP`.

2.5.2 CoViNE-ILP: ILP formulation for CoViNE Embedding

CoViNE-ILP, takes a k-protected VN $\hat{G}$ and its conflicting set $\chi^{\hat{G}}$ as inputs. It embeds $\hat{G}$ on an SN G while ensuring the disjointness constraints imposed by $\chi^{\hat{G}}$ and minimizing cost according to (2.17). *CoViNE-ILP* is similar to *CoViNE-opt*, excluding the variables and constraints related to the augmentation and disjointness requirement computation. *CoViNE-ILP* does not use the augmentation variable (*i.e.*, $a_k^{\bar{u}\bar{v}}$) and constraint (2.2) of *CoViNE-opt* since augmentation is performed by Alg. 1. Therefore, *CoViNE-ILP* replaces (2.4) by the following:

$$\begin{aligned} &\forall(\bar{u},\bar{v})\in\bar{E},\forall k\in K,\forall u\in V: \\ &\sum_{\forall v\in\mathcal{N}(u)}(x_{uv}^{\bar{u}\bar{v}k}-x_{vu}^{\bar{u}\bar{v}k})=y_{\bar{u}u}-y_{\bar{v}u} \end{aligned} \tag{2.18}$$

Furthermore, *CoViNE-ILP* uses the disjointness constraints imposed by pre-computed $\chi^{\hat{G}}$. Hence, it eliminates disjoint group assignment variable $d_{\bar{C}_i}^{\bar{u}\bar{v}k}$ and replaces disjointness

constraints (2.12) and (2.13) from *CoViNE-opt* with the following:

$$\begin{aligned} &\forall (u,v) \in E, \forall (\bar{u},\bar{v}) \in \bar{E}, \forall k \in K, \forall (\bar{a},\bar{b})^q \in \chi^{\bar{u}\bar{v}k} : \\ &x_{uv}^{\bar{u}\bar{v}k} + x_{vu}^{\bar{u}\bar{v}k} + x_{uv}^{\bar{a}\bar{b}q} + x_{vu}^{\bar{a}\bar{b}q} \leq 1 \end{aligned} \tag{2.19}$$

(2.19) ensures that VLink embedding of $(\hat{u},\hat{v})^k \in \hat{E}$ will never share an SLink with the embeddings of the conflicting VLinks present in $\chi^{\hat{u}\hat{v}k}$, thus satisfying the disjointness relation among them. If the SN does not have sufficient number of EDSPs to satisfy the disjointness constraint (2.19), *CoViNE-ILP* becomes infeasible rendering no solution for the embedding. The total number of binary variables for $x_{uv}^{\bar{u}\bar{v}k}$ is $|\bar{E}| \times k \times |E|$ and total number of constraints generated by (2.19) is $k \times |E| \times |\bar{E}|^2$. Therefore, unlike *CoViNE-opt*, which has an exponential number of variables and constraints, *CoViNE-ILP* has a polynomial number of variables and constraints. Note that even with the polynomial number of variables and constraints *CoViNE-ILP* is intractable as the worst case time complexity is $O((k \times |E| \times |\bar{E}|^2) \times 2^{(|\bar{E}| \times k \times |E|)})$.

2.5.3 Heuristic Algorithm for CoViNE Embedding

Alg. 2 computes two functions, *nmap* and *emap*, which represent the VNode and VLink mapping of $\hat{G}$ on G, respectively. Since there is no cost associated with VNode mapping, a VLink mapping that minimizes total cost determines the VNode mapping. As discussed in § 2.4.1, disjointness constraints imposed by conflicting sets may lead to infeasible solutions. Hence, Alg. 2 prioritizes finding a feasible mapping than minimizing the embedding cost. Following this intuition, Alg. 2 first sorts the VNodes $\hat{u} \in \hat{V}$ in decreasing order of the sum of conflicting set sizes of incident VLinks. This sorted list of VNodes is represented by $\hat{\mathcal{V}}$. Since a VNode with VLinks having larger conflicting sets becomes too constrained to be mapped to a suitable SNode, Alg. 2 tries to map VNodes in the order of $\hat{\mathcal{V}}$. However, Alg. 2 maps a VNode from $\hat{\mathcal{V}}$ to the candidate SNode that satisfies the disjointness constraints and minimizes the embedding cost.

For each VNode $\hat{u} \in \hat{\mathcal{V}}$, Alg. 2 searches for an unallocated SNode in $\hat{u}$'s location constraint set, $L(\hat{u})$, which yields a feasible mapping with the minimum cost. To embed $\hat{u}$, Alg. 2 loops through each candidate SNode $l \in L(\hat{u})$ (Line $5-13$), to first temporarily map $\hat{u}$ to l (Line 6). Then the algorithm computes temporary mappings for all the VLinks incident to $\hat{u}$. The `VL-MAP` (Alg. 3) procedure is invoked to find the mapping for each such VLink (Line 9). VLinks incident to $\hat{u}$ are picked in decreasing order of their conflicting set sizes to maximize the chances of finding a feasible solution. Alg. 2 finally embeds $\hat{u}$ to the candidate l that leads to a feasible mapping for all the VLinks incident to $\hat{u}$ and yields

Algorithm 2: Compute VN Embedding

1 **function** *VN-EMBEDDING*$(G, \hat{G}, \chi^{\hat{G}})$
2 $\quad \hat{\mathcal{V}} \leftarrow$ Sort $\hat{u} \in \hat{V}$ in decreasing order of $\sum_{\forall \hat{v} \in \mathcal{N}(\hat{u})} \sum_{\forall k \in K} |\chi^{\hat{u}\hat{v}k}|$
3 $\quad$ **foreach** $\hat{u} \in \hat{\mathcal{V}}$ **do**
4 $\quad\quad Candidate \leftarrow \phi$
5 $\quad\quad$ **foreach** $l \in L(\hat{u})$ **do**
6 $\quad\quad\quad nmap(\hat{u}) \leftarrow l$
7 $\quad\quad\quad \mathcal{E} \leftarrow$ Sort $(\hat{u}, \hat{v})^q \in \hat{E}$ in decreasing order of $|\chi^{\hat{u}\hat{v}q}|$
8 $\quad\quad\quad$ **foreach** $(\hat{u}, \hat{v})^q \in \mathcal{E} |\ emap(\hat{u}, \hat{v})^q = \phi$ **do**
9 $\quad\quad\quad\quad P[(\hat{u}, \hat{v})^q] \leftarrow$ VL-MAP$(G, \hat{G}, \chi^{\hat{G}}, (\hat{u}, \hat{v})^q)$
10 $\quad\quad\quad$ **if** $\sum_{\forall (\hat{u}, \hat{v})^q \in \mathcal{E}} cost(P[(\hat{u}, \hat{v})^q])$ *is minimum* **then**
11 $\quad\quad\quad\quad M \leftarrow P$, $Candidate \leftarrow l$
12 $\quad\quad\quad nmap(\hat{u}) \leftarrow \phi$
13 $\quad\quad\quad \forall (\hat{u}, \hat{v})^q \in \mathcal{E}$: $emap(\hat{u}, \hat{v})^q \leftarrow \phi$
14 $\quad\quad$ **if** $Candidate \neq \phi$ **then**
15 $\quad\quad\quad$ Add mapping $\hat{u} \rightarrow Candidate$ to $nmap$
16 $\quad\quad\quad \forall (\hat{u}, \hat{v})^q \in \mathcal{E} | nmap(\hat{u}) \neq \phi \wedge nmap(\hat{v}) \neq \phi$:
17 $\quad\quad\quad\quad$ Add $(\hat{u}, \hat{v})^q \rightarrow M[(\hat{u}, \hat{v})^q]$ to $emap$
18 $\quad\quad$ **else return** No Solution Found
19 $\quad$ **return** $\{nmap, emap\}$

the minimum embedding cost (Line 15). The algorithm fails, if no such feasible l is found. Once a VNode $\hat{u}$ has been finally mapped, Alg. 2 creates the final mapping for only those VLinks incident to $\hat{u}$ whose both endpoints are already finally mapped (Line 17). If we map a VLink with one unmapped endpoint (*e.g.*, , $\hat{v}$), we have to map $\hat{v}$ based on this local information. This would reduce the degrees of freedom for $\hat{v}$ when other VLinks incident to $\hat{v}$ would have been mapped, and may lead to infeasible solution. Mappings of such VLinks incident to $\hat{u}$ are finalized when their unmapped endpoints are finally mapped.

We now describe the `VL-MAP` (Alg. 3) procedure for finding the mapping of a VLink, $(\hat{u}, \hat{v})^q$. First we remove all the SLinks used by the mappings of all the VLinks in $\chi^{\hat{u}\hat{v}q}$ to satisfy the disjointness constraints (Line 4). Then, we compute mapping for $(\hat{u}, \hat{v})^q$ by considering the following two cases: (i) both endpoints of $(\hat{u},\ \hat{v})^q$ have already been mapped to some SNodes. In this case, we find a minimum cost path between $nmap(\hat{u})$

Algorithm 3: Compute VLink Mapping

1 **function** *VL-MAP*$(G, \hat{G}, \chi^{\hat{G}}, (\hat{u}, \hat{v})^q)$
2 $\quad p^{\hat{u}\hat{v}} \leftarrow \phi$
3 $\quad$ **foreach** $(\hat{s}, \hat{t})^r \in \chi^{\hat{u}\hat{v}q}$: **do**
4 $\quad\quad E \leftarrow E - \{(a, b) \in E | (\hat{s}, \hat{t})^r$ is mapped to $(a, b)\}$
5 $\quad$ **if** $nmap(\hat{u}) \neq \phi \wedge nmap(\hat{v}) \neq \phi$ **then**
6 $\quad\quad Q^{nmap(\hat{u})nmap(\hat{v})} \leftarrow \text{MP}(G, nmap(\hat{u}), nmap(\hat{v}), b_{\hat{u}\hat{v}})$
7 $\quad$ **else if** $nmap(\hat{u}) = \phi \wedge nmap(\hat{v}) \neq \phi$ **then**
8 $\quad\quad Q^{nmap(\hat{u})nmap(\hat{v})} \leftarrow \min_{\forall l \in L(\hat{u})} \{\text{MP}(G, l, nmap(\hat{v}), b_{\hat{u}\hat{v}})\}$
9 $\quad$ **else if** $nmap(\hat{u}) \neq \phi \wedge nmap(\hat{v}) = \phi$ **then**
10 $\quad\quad Q^{nmap(\hat{u})nmap(\hat{v})} \leftarrow \min_{\forall l \in L(\hat{v})} \{\text{MP}(G, nmap(\hat{u}), l, b_{\hat{u}\hat{v}})\}$
11 $\quad$ **if** $Q^{nmap(\hat{u})nmap(\hat{v})} \neq \phi$ **then**
12 $\quad\quad$ Add $(\hat{u}, \hat{v})^q \rightarrow Q^{nmap(\hat{u})nmap(\hat{v})}$ to *emap*
13 $\quad$ **return** $Q^{nmap(\hat{u})nmap(\hat{v})}$

and $nmap(\hat{v})$ with capacity at least $b_{\hat{u}\hat{v}}$ in G (Line 6); (ii) only $\hat{u}$ (or $\hat{v}$) is mapped and the other endpoint $\hat{v}$ (or $\hat{u}$) has not been mapped. In this case, we compute the minimum cost path between $nmap(\hat{u})$ (or $nmap(\hat{v})$) and all possible locations for the unmapped VNode $\hat{v}$ (or $\hat{u}$), $l \in L(\hat{v})$ (or $L(\hat{u})$) with at least $b_{\hat{u}\hat{v}}$ capacity (Line 8 (or Line 10)). The VLink $(\hat{u}, \hat{v})^q$ is temporarily mapped to the computed path and the mapping is added to *emap* (Line 12). We modified *Dijkstra's shortest path* algorithm [53] to consider SLink capacities, while computing the minimum-cost path (`MP` procedure call in Alg. 3). The cost of each SLink $(u, v) \in E$ is set to $C_{uv} \times b_{\hat{u}\hat{v}}$, where $b_{\hat{u}\hat{v}}$ is the bandwidth requirement of the VLink to be embedded.

Discussion

The most computationally expensive step of Alg. 2 is the `VL-MAP` procedure, which invokes *Dijkstra's shortest path* algorithm requiring $O(|E| + |V| \log |V|)$ time. Since `VL-MAP` is invoked $O(|\hat{V}||L(\hat{u})||\mathcal{N}(\hat{u})|)$ times, the running time of Alg. 2 is $O(|\hat{V}||L(\hat{u})||\mathcal{N}(\hat{u})|(|E| + |V| \log |V|))$. Unlike most of the approaches in the literature that perform VNode and VLink mapping sequentially [59], Alg. 2 addresses VNode mapping and disjointness constrained VLink mapping simultaneously. However, Alg. 2 maps VNodes of a VN (or, the VLinks incident to a VNode) one-by-one, starting from the most constrained VNode (or, VLink) to the least constrained one. Although these orders increase the chances of finding

a feasible solution, they may lead to sub-optimal solutions. Meta-heuristic approaches can achieve better orders by exploring larger solution space at the cost of increased execution time [21].

2.6 Evaluation

We evaluate our proposed solutions for *CoViNE* through extensive simulations. We briefly discuss the compared approaches in § 2.6.1, performance metrics in § 2.6.2 followed by the simulation setup in § 2.6.3. Finally, we describe our evaluation results in § 2.6.4 focusing on optimality, scalability, embedding performance, and failure restoration.

2.6.1 Compared Approaches

We compare the performance of our sub-optimal solutions (*CoViNE-ILP* and *CoViNE-fast*) to the optimal solution, *CoViNE-opt* under single ($k = 1$) and double ($k = 2$) SLink failure scenarios. We have restricted our failure scenarios to double link failures, since the possibility of more than two simultaneous failures is extremely low in practice [117, 65]. To measure how much extra resources are needed to guarantee connectivity under different failure scenarios, we compare our approaches with an optimal VNE algorithm [185] that does not ensure any connectivity during substrate failure ($k = 0$). Table 2.2 summarizes these approaches by listing the failure scenarios and names of the ILP and algorithm they use along with their worst case time complexities.

2.6.2 Performance Metrics

Embedding Cost

The cost of provisioning bandwidth for the VLinks in a VN, computed using (2.17).

Execution Time

The time required for an algorithm to find the solution for CoViNE.

Table 2.2: Compared Approaches

Failure Scenario	**Notation**	**Augmentation & conflicting set computation**	**Embedding**	**Worst case time complexity**
$k = 0$	ViNE-ILP [185]	None	ILP of MCUFP	$O((\lvert E\rvert \times \lvert\bar{E}\rvert) \times 2^{(\lvert\bar{E}\rvert\times\lvert E\rvert)})$
$k = 1$	S-CoViNE-opt	CoViNE-opt	CoViNE-opt	$O((\lvert E\rvert \times (2^{\lvert\bar{V}\rvert} - 2)) \times 2^{(\lvert\bar{E}\rvert\times(2^{\lvert\bar{V}\rvert}-2))})$
	S-CoViNE-ILP	Alg. 1	CoViNE-ILP	$O((\lvert E\rvert \times \lvert E\rvert^2) \times 2^{(\lvert\bar{E}\rvert\times\lvert E\rvert)})$
	S-CoViNE-fast	Alg. 1	Alg. 2	$O(2 \times \lvert V\rvert \times \lvert\mathcal{N}(\bar{u})\rvert \times \lvert L(\bar{u})\rvert^2 \times (\lvert E\rvert + \lvert V\rvert \times \log\lvert V\rvert))$
$k = 2$	D-CoViNE-opt	CoViNE-opt	CoViNE-opt	$O((\lvert E\rvert \times (2^{\lvert\bar{V}\rvert} - 2)) \times 2^{(\lvert\bar{E}\rvert\times 2\times(2^{\lvert\bar{V}\rvert}-2))})$
	D-CoViNE-ILP	Alg. 1	CoViNE-ILP	$O((2 \times \lvert E\rvert \times \lvert E\rvert^2) \times 2^{(\lvert\bar{E}\rvert\times 2\times\lvert E\rvert)})$
	D-CoViNE-fast	Alg. 1	Alg. 2	$O(3 \times \lvert V\rvert \times \lvert\mathcal{N}(\bar{u})\rvert \times \lvert L(\bar{u})\rvert^2 \times (\lvert E\rvert + \lvert V\rvert \times \log\lvert V\rvert))$

Restored Bandwidth

The percentage of VLinks' bandwidth that is restored after these VLinks have been affected by one or more SLink failure(s).

2.6.3 Simulation Setup

We implement the ILP formulations of *CoViNE-opt* and *CoViNE-ILP* using IBM ILOG CPLEX C++ library and Alg. 1 and Alg. 2 using C++. The evaluation was performed on a machine with 8×10-core hyper-threaded Intel Xeon E7-8870 2.40GHz CPU and 2TB RAM. To demonstrate the scalability of our solutions, we consider both small and large network topologies summarized in Table 2.3. For small and large cases, we vary both

the size and LNR of SNs and VNs to assess the robustness of our solutions. Note that the problem instances have been selected by studying ISP network measurement research literature [153, 125, 100] and consulting with our industry partners. For each problem instance in Table 2.3, with a given SN size and VN size, we generate 3 VNs for each generated SN, and execute the compared approaches in Table 2.2 to embed each VN on the SN independently. For a particular VN and SN, the source and destination of an SLink or a VLink and the location constraints of VNodes are chosen randomly, and VLink demands are set to 10% of the SLink bandwidths. For each VN, we measure the performance metrics and plot the metrics' average value with errorbar showing the maximum and the minimum values. To analyze the impact of using different algorithms for solving VN embedding (Fig. 2.6), we use star, ring, and randomly connected VN topologies on anonymous inter-continental network topologies with varying Link-to-Node Ratios (LNRs). However, due to huge cost of deploying inter-continental links, these SNs do not have enough LNRs to guarantee necessary edge-connectivity against double link failures. Therefore, we do not evaluate double link failure scenarios for this analysis. In addition, we demonstrate how our approaches can survive affected VLinks' bandwidth in the presence of single and double SLink failures (Fig. 2.9).

Table 2.3: Summary of Simulation Parameters

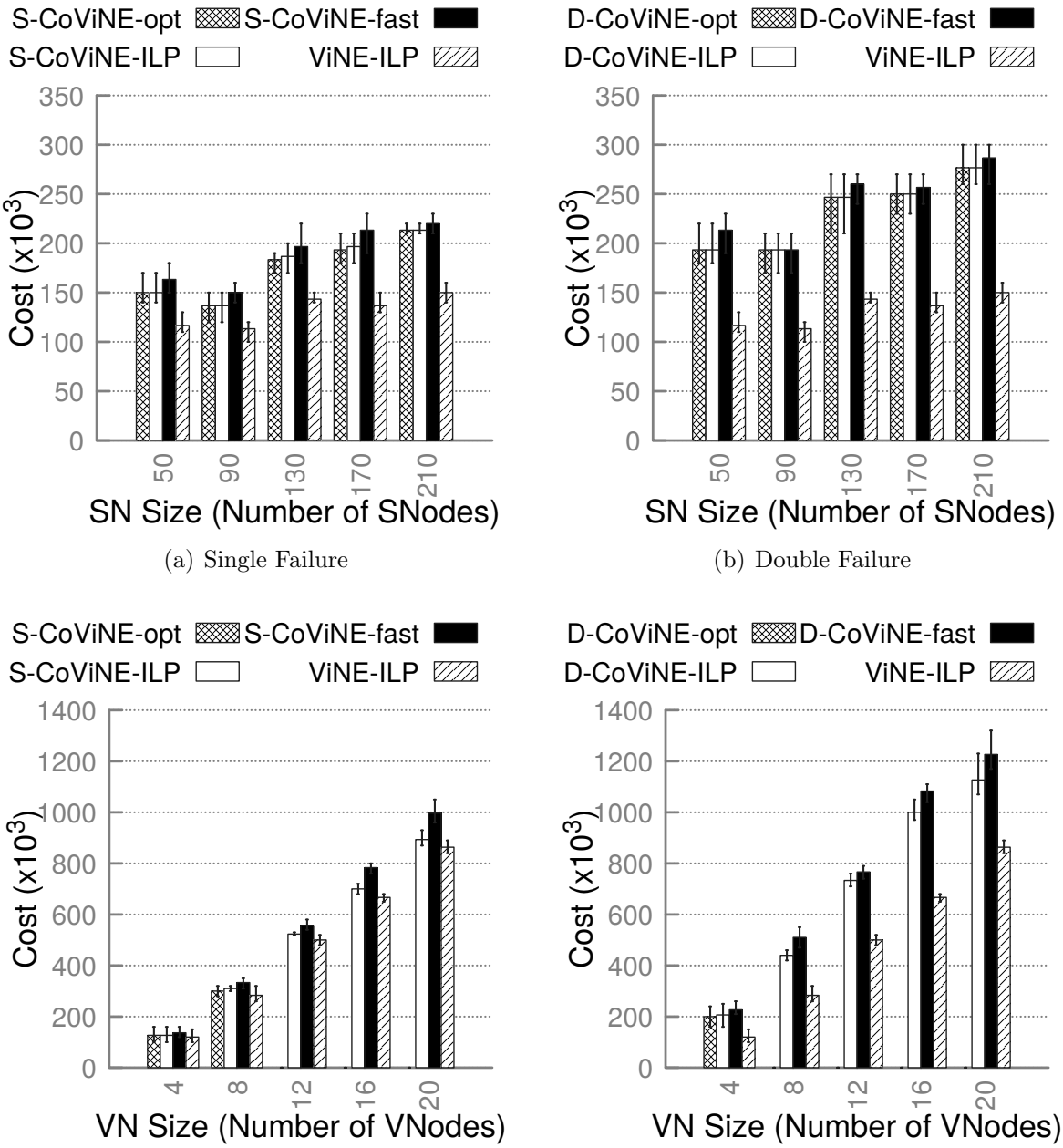
Scenario	**Figure(s)**	**SNodes**	**SLinks**	**VNodes**	**VLinks**
Small Scale	2.3(a), 2.3(b), 2.5(a), 2.5(b)	50-210	105-412	5	7
	2.3(c), 2.3(d), 2.5(c), 2.5(d)	100	198	4-20	5-37
	2.4(a), 2.4(b)	25	38-78	5	7
	2.4(c), 2.4(d)	25	46	6	6-14
	2.6	29-158	35-303	3-20	3-40
Large Scale	2.7(a)	500	2017	10-100	21-285
	2.8(a)	1000	4023	10-100	21-285
	2.7(b)	500	1500	10	11-31
	2.8(b)	1000	2000	10	11-31
	2.7(c)	500	1000-2000	10	21
	2.8(c)	1000	2000-4000	10	21
Failure Restoration	2.9	150	310	10	11-31

2.6.4 Results

Small Scale Scenarios

Optimality Analysis Fig. 2.3(a)–2.3(d) compares embedding cost for guaranteeing connectivity against different failure scenarios ($k = 0, 1$, and 2) by varying SN and VN sizes, while Fig. 2.4(a)–2.4(d) presents embedding cost by varying SN and VN LNRs. Fig. 2.3(a) and Fig. 2.3(b) show that cost increases for all the compared approaches with the increase in SN size. This is due to the fact that location constraints of the VNodes of a VN are placed far apart from one another in a larger SN, thus involving more SLinks in the substrate paths for mapping VLinks of the VN and consuming more substrate resources along those SLinks. An opposite trend is observed when SN LNR is increased in Fig. 2.4(a) and Fig. 2.4(b). Incrementing SN density increases substrate path diversity with higher number of shorter and disjoint paths that result in lower embedding costs. In constrast, increasing VN size or VN LNR escalates embedding cost in general as seen in Fig. 2.3(c), Fig. 2.3(d), and Fig. 2.4(c) except for double link failure scenario in Fig. 2.4(d). This stems from the fact that a larger or denser VN has a higher number of VLinks to be embedded that increase substrate resource consumption. In contrast, sparse VNs require higher number of VLinks to be augmented (see Fig. 2.9(d)) to guarantee connectivity against double link failures that results in higher cost for embedding the augmented VLinks at lower VN LNRs.

Among the approaches that guarantee connectivity for single failure scenario (*i.e.*, $k = 1$) as shown in Fig. 2.3(a) and Fig. 2.3(c), *S-CoViNE-opt* generates the lowest cost of embedding. Such behavior is expected since *S-CoViNE-opt* jointly optimizes all the sub-problems of *CoViNE* yielding the optimal cost, whereas *S-CoViNE-ILP* and *S-CoViNE-fast* address them sequentially. However, as can be seen in Fig. 2.3(c), *S-CoViNE-opt* does not scale beyond a VN of size 8 due to its dependency on the exponential number of edge-cuts in the VN. *S-CoViNE-ILP* eliminates this dependency by leveraging conflicting set of a VN and adopting a heuristic algorithm for performing augmentation and computing disjointness requirement. Despite using a heuristic for solving part of CoViNE, the embedding costs of *S-CoViNE-ILP* remain very close (within $\sim$ 3% on average) to those of *S-CoViNE-opt*. In contrast, *S-CoViNE-fast* employs heuristic algorithms for all the sub-problems of *CoViNE* and incurs $\sim$18% and $\sim$16% additional cost compared to *S-CoViNE-opt* and *S-CoViNE-ILP*, respectively. Slightly higher optimality gaps are observed among the solutions for double failure scenarios (*i.e.*, $k = 2$) in Fig. 2.3(b) and Fig. 2.3(d) compared to single failure cases. In particular, the optimality gap between *D-CoViNE-ILP* and *D-CoViNE-opt* is $\sim$ 4%, whereas the same between *D-CoViNE-fast* and *D-CoViNE-ILP* is $\sim$ 19%. The increased optimality gaps among the solutions for



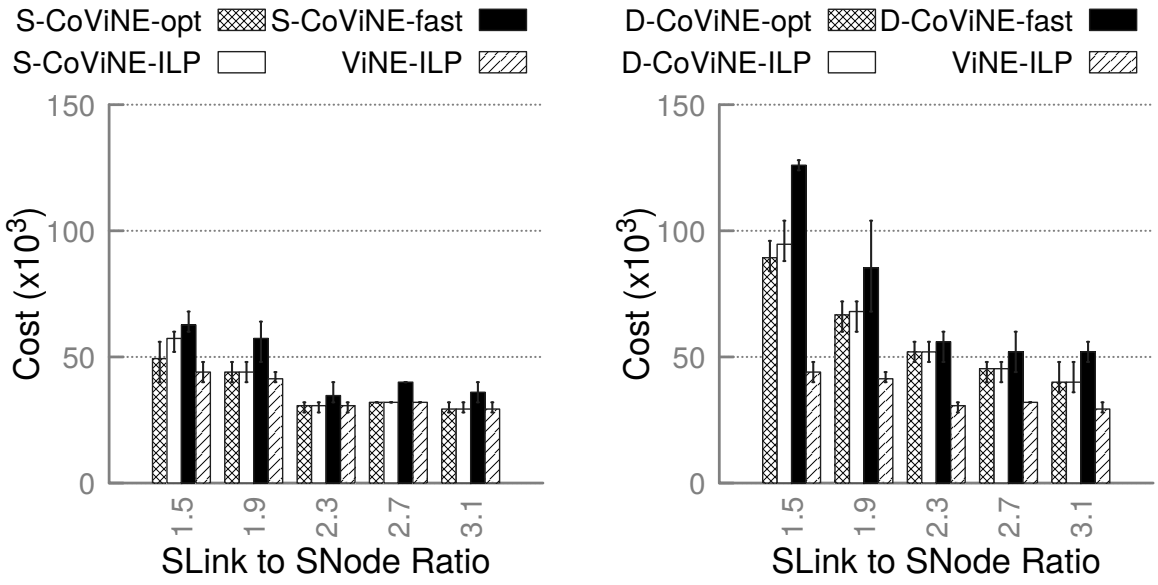
(a) Single Failure

(b) Double Failure

(c) Single Failure

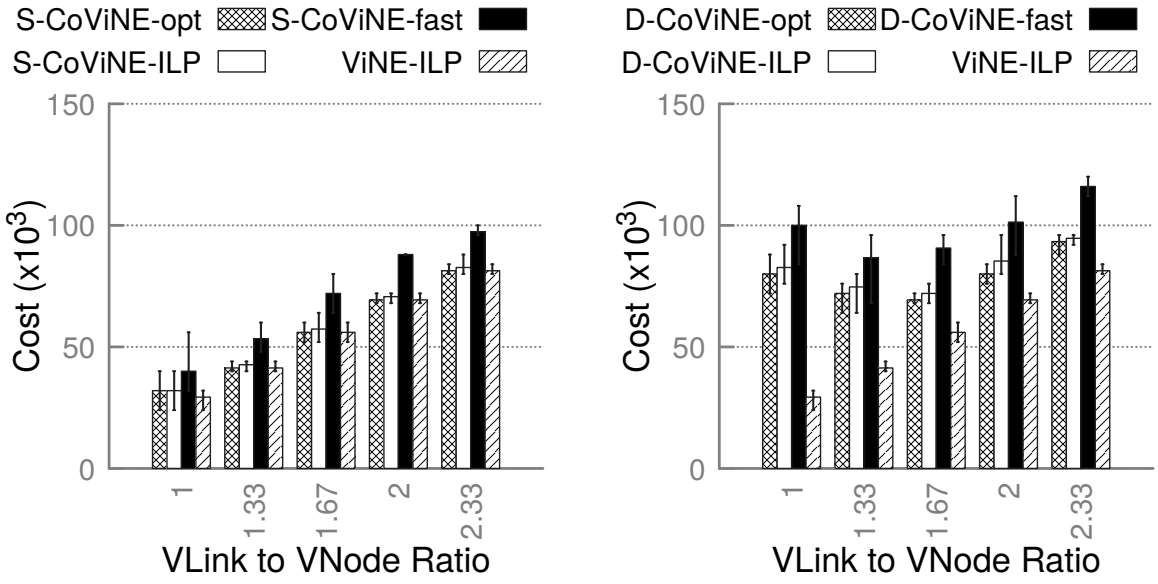
(d) Double Failure

Figure 2.3: Embedding Cost Analysis by varying topology size for Small Scale Topologies



(a) Single Failure

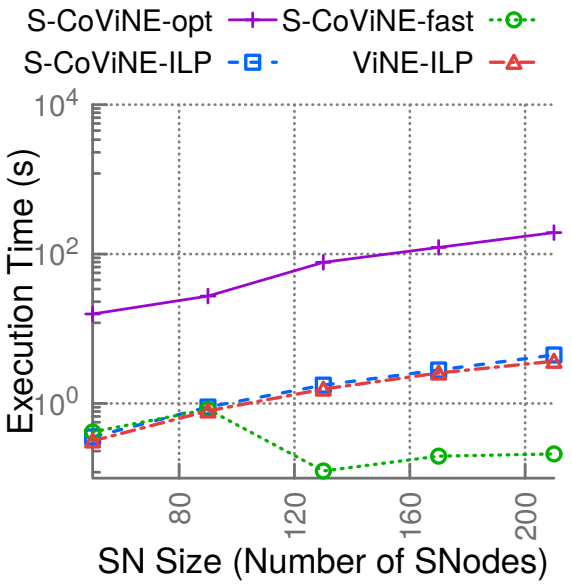
(b) Double Failure



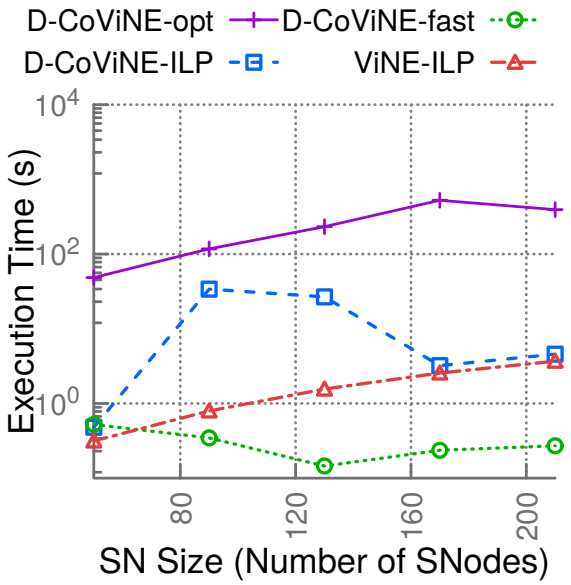
(c) Single Failure

(d) Double Failure

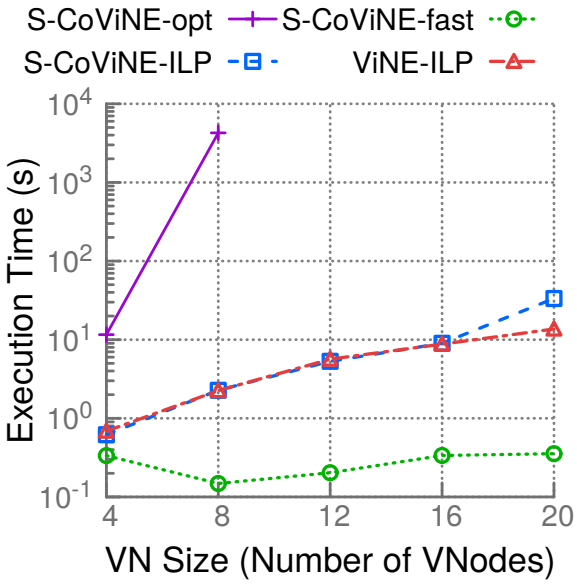
Figure 2.4: Embedding Cost Analysis by varying topology LNR for Small Scale Topologies



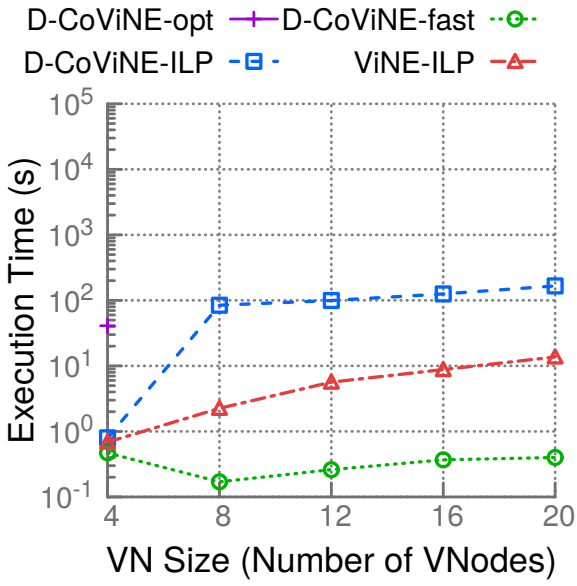
(a) Single Failure



(b) Double Failure



(c) Single Failure



(d) Double Failure

Figure 2.5: Execution Time Analysis for Small Scale Topologies

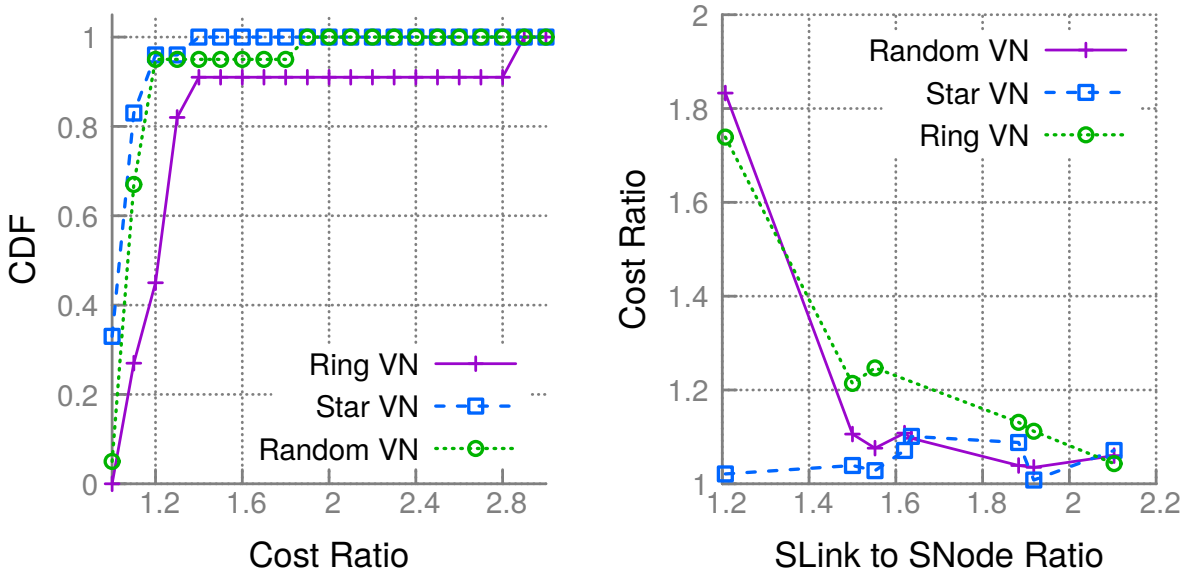
$k = 2$ is due to having more augmentation and disjointness constraints than the single failure solutions.

Embedding Cost for Different Failure Scenarios A comparison of costs between single and double failure scenarios of any solution (*e.g.*, *S-CoViNE-opt* and *D-CoViNE-opt*) in Fig. 2.3(a)–2.3(d) reveals that ensuring connectivity against a higher degree of failure ($k = 2$) incurs a higher embedding cost. The reasons are twofold. First, *D-CoViNE-opt*, as well as *D-CoViNE-ILP* and *D-CoViNE-fast*, require more parallel VLinks to be added than their single failure counterparts to ensure the necessary edge-connectivity in the VN, consuming additional substrate resources for embedding the augmented VLinks. Second, solutions for double SLink failures require more disjointness constraints to be satisfied to preserve necessary edge-connectivity in the embedding of the VN, resulting in longer substrate paths for VLink mapping. Following these arguments, *ViNE-ILP* (with $k = 0$) produces the lowest cost of embedding, since it neither augments any VLink nor imposes any disjointness constraint. Empirically, *D-CoViNE-opt* incurs ∼24% and ∼66% more cost than *S-CoViNE-opt* and *ViNE-ILP*, respectively.

Scalability Analysis Fig. 2.5(a)–2.5(d) reports execution times in logarithmic scale for guaranteeing connectivity against different failure scenarios by varying the SN and VN sizes. These figures show that the execution times of all the approaches, relying on ILP formulation for part of the problem, increase exponentially with increasing problem size. In contrast, execution times of the heuristic (*i.e.*, *S-CoViNE-fast* and *D-CoViNE-fast*) remain well within a second for similar problem instances. Furthermore, *S-CoViNE-opt* and *D-CoViNE-opt* are the slowest among the approaches for single and double failure scenarios, respectively, due to the intricacy of *CoViNE-opt* as discussed in § 2.3.4. In addition, *D-CoViNE-opt* is an order of magnitude slower than *S-CoViNE-opt* due to having more disjointness constraints and variables than *S-CoViNE-opt*. Finally, Fig. 2.5(c) and Fig. 2.5(d) reveal that VN size has a more profound impact on the scalability of the ILP based approaches than what SN size has. For instance, with our current hardware, *CoViNE-opt* and *CoViNE-ILP* hit a ceiling in terms of VN size of 8 and 22 nodes, respectively, on a 100 node SN.

Optimality Analysis of Alg. 2 We now analyze how much extra resources are allocated by *S-CoViNE-fast* compared to *S-CoViNE-ILP* for different VN topologies. This extra resource usage is measured as the ratio of their embedding costs since both *S-CoViNE-fast* and *S-CoViNE-ILP* use the same Alg. 1 for augmentation and conflicting set computation.

Therefore, the cost ratio indicates the perfomance comparison between ILP formulation for VN embedding and Alg. 2. Fig. 2.6(a) presents the Cumulative Distribution Function (CDF) of cost ratio for different types of VNs. This plot shows that 95% VNs with star and random topologies are embedded by *S-CoViNE-fast* with at most 20% extra resources compared to *S-CoViNE-ILP*. For ring topologies, costs of 91% of the VNs incurred by *S-CoViNE-fast* remain within 35% of those of *S-CoViNE-ILP*. Ring topologies are different from star and random topologies as all the VLinks in the ring need to be embedded on mutually disjoint substrate paths to ensure connectivity against failures. Such stringent disjointness constraints affect the sequential VLink mapping of Alg. 2 in *S-CoViNE-fast*, resulting in the highest cost ratio for ring topologies. However, the higher costs of *S-CoViNE-fast*, compared to *S-CoViNE-ILP*, are compensated by their higher scalability and much faster execution time as discussed earlier.



(a) CDF of Cost Ratio

(b) Impact of SN connectivity

Figure 2.6: Performance Analysis of Alg. 2

Fig. 2.6(a) exhibits higher cost ratios between *S-CoViNE-fast* and *S-CoViNE-ILP* than those observed in Fig. 2.3(a). This is due to using real SN topologies that are sparser than synthetic SNs used in Fig. 2.3(a). To analyze this further, Fig. 2.6(b) shows how the density of the underlying SN impacts the performance of *S-CoViNE-fast*. As we observe in Fig. 2.6(b), the costs of *S-CoViNE-fast* and *S-CoViNE-ILP* for ring and random VNs differ

by a larger margin in a ring-like SN (*i.e.*, LNR=1.2). In this extreme case, the penalty for missing an optimal solution by *S-CoViNE-fast* is substantial as the disjoint path becomes much longer than the shorter one due to lack of SN path diversity, resulting in a higher cost ratio. However, cost ratios between *S-CoViNE-fast* and *S-CoViNE-ILP* decrease initially with the increase in SN LNR (Fig. 2.6(b)), and do not change significantly for SNs with LNR$\geq$1.6. An increase in SN LNR increases SN path diversity. *S-CoViNE-fast* exploits this path diversity to find a shorter path for embedding a VLink, resulting in lower cost ratios. For star VNs, disjointness requirement is much less than denser VNs, yielding close to unit cost ratios in all cases.

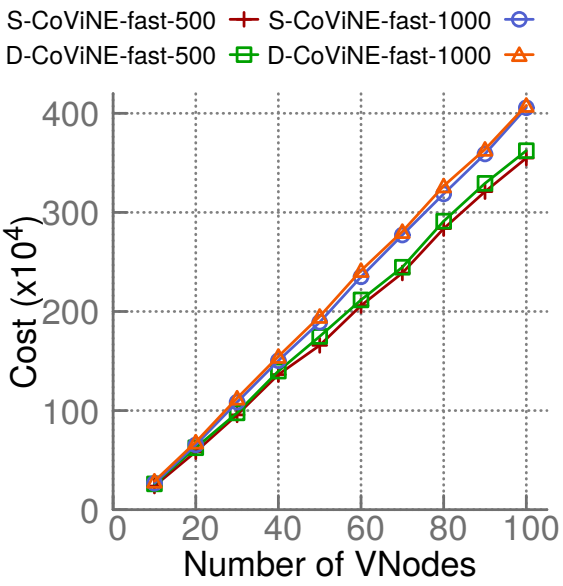
Large Scale Scenarios

Embedding Cost Fig. 2.7(a) shows embedding costs of *S-CoViNE-fast* and *D-CoViNE-fast* with varying VN sizes on 500 and 1000 node SNs. As expected, cost increases with the increase in both VN size and SN size. Fig. 2.7(b) and Fig. 2.7(c) show embedding cost by varying VN and SN LNRs, respectively. In these scenarios, embedding cost is mostly influenced by disjointness constraint and parallel VLink augmentation. For *D-CoViNE-fast*, augmentation cost dominates for VNs with LNR $\leq$ 2.1 (see Fig. 2.9(d)). In addition, the number of augmented VLinks decreases with the increase in VN LNR, hence the initial decrease in embedding cost. However for VNs with LNR $>$ 2.1, cost for ensuring disjointness constraint dominates, which justifies the later increase in Fig. 2.7(b) for *D-CoViNE-fast*. On the other hand, for *S-CoViNE-fast*, disjointness constraint dominates and embedding cost increases as higher number of VLinks are embedded on the same SN for VNs with larger LNR. For the same reason discussed for small cases, higher path diversity accounts for the decrease in cost with an increase in SN LNR in Fig. 2.7(c).

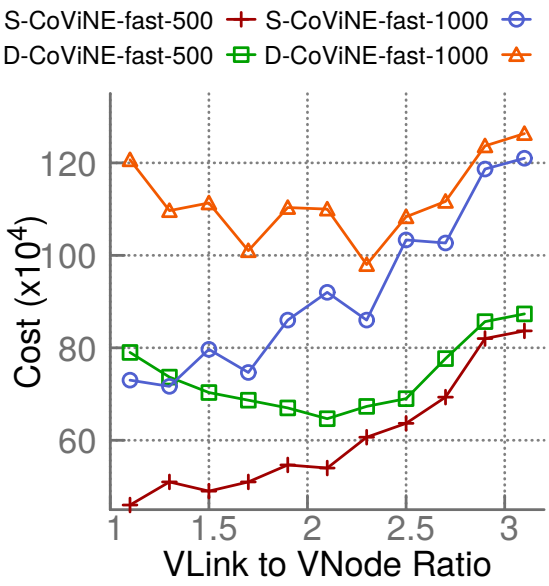
Scalability Analysis Confirming to the running time analysis in § 2.5.1 and § 2.5.3, the execution times for *S-CoViNE-fast* and *D-CoViNE-fast* increase with the increase in VN and SN sizes (Fig. 2.8(a)), VN LNR (Fig. 2.8(b)), and SN LNR (Fig. 2.8(c)). In addition, the execution times for *S-CoViNE-fast* and *D-CoViNE-fast* are comparable with each other in the large scale scenarios.

2.6.5 Failure Restoration

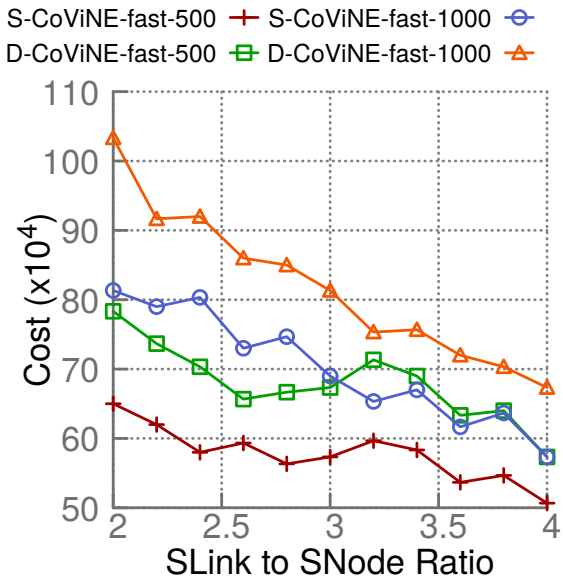
To demonstrate the failure restoration capability of CoViNE, we steer three classes of traffic in a VN, labeled as Pr-1 (highest priority), Pr-2, and Pr-3 (lowest priority) demanding 20%,



(a) Cost Vs. VN Size

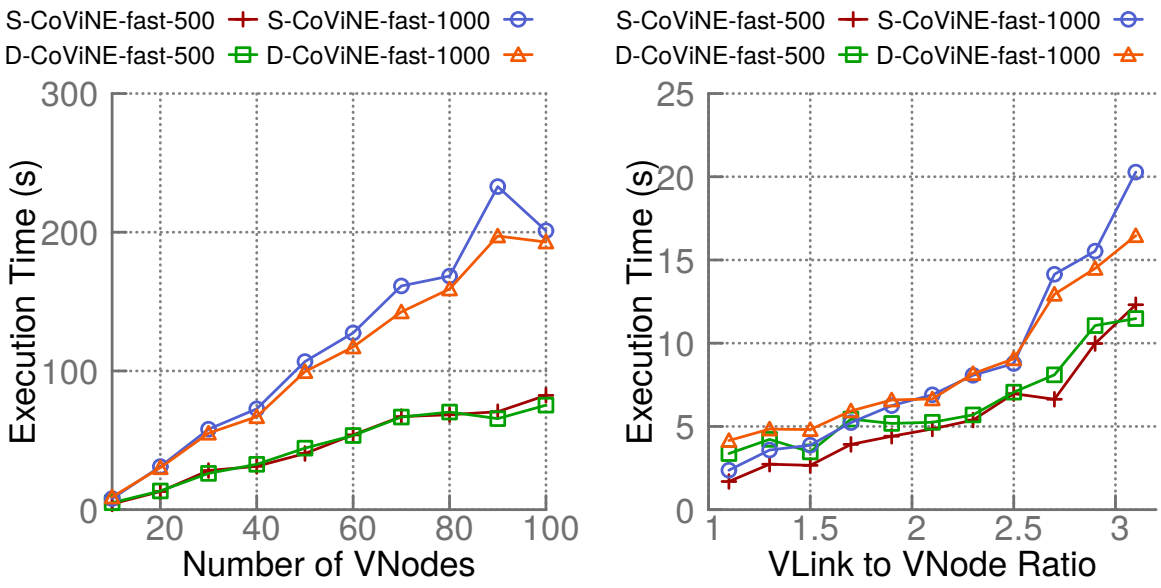


(b) Cost Vs. VN LNR



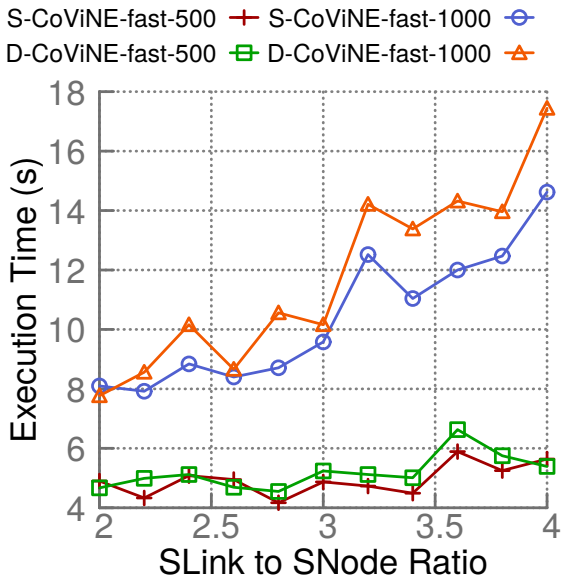
(c) Cost Vs. SN LNR

Figure 2.7: Embedding Cost Analysis for Large Scale Topologies



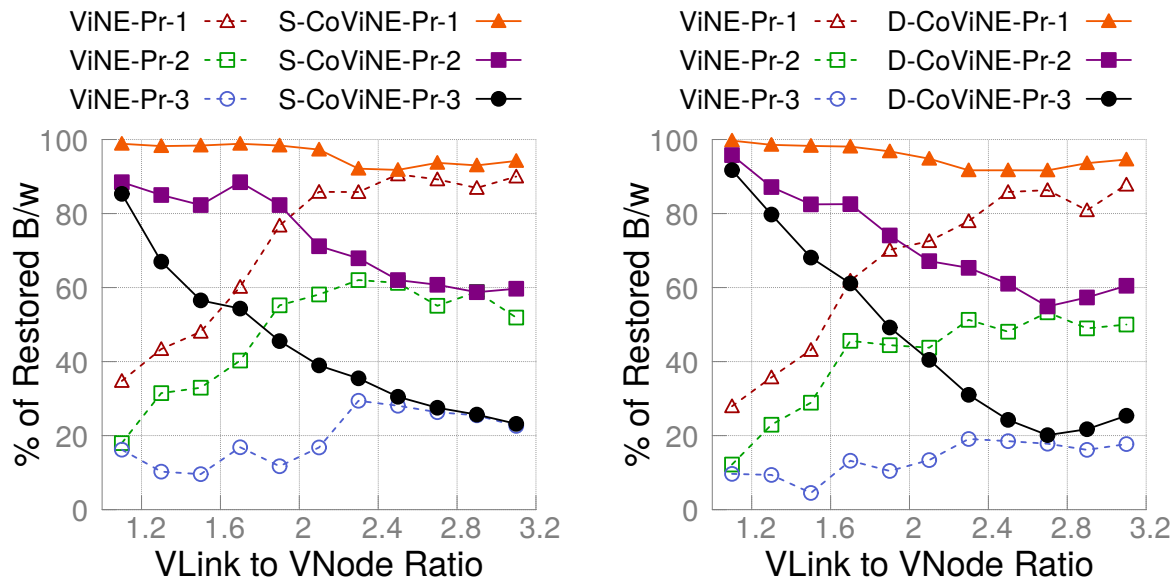
(a) Time Vs. VN Size

(b) Time Vs. VN LNR



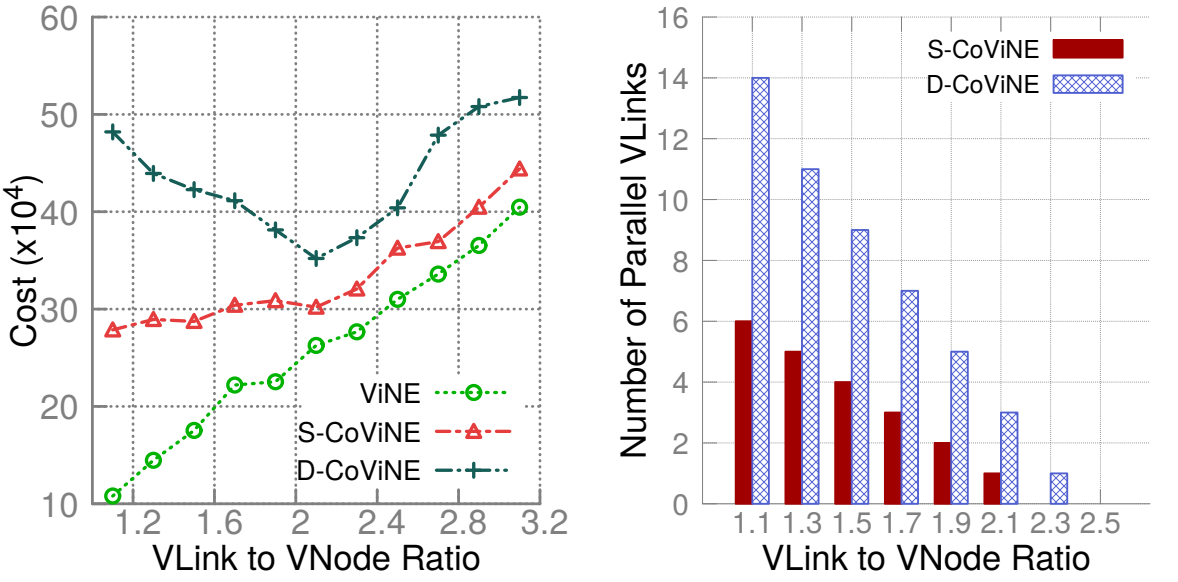
(c) Time Vs. SN LNR

Figure 2.8: Execution Time Analysis for Large Scale Topologies



(a) Single Failure

(b) Double Failure



(c) Embedding Cost Overhead

(d) VLink Aug. Overhead

Figure 2.9: Restored Bandwidth and Overhead Analysis

30%, and 50% of each VLink's bandwidth, respectively. A controller handles failures by rerouting traffic in the affected VLinks along alternate shortest paths in the embedding of the VN. Bandwidth sharing along these paths follows fair sharing policy between traffic from the same class and weighted fair sharing across different traffic classes.

Fig. 2.9(a) and Fig. 2.9(b) show the restored bandwidth of *CoViNE-fast* and ViNE-ILP in the presence of single and double SLink failures, respectively. On the other hand, Fig. 2.9(c) and Fig. 2.9(d) present the overhead for ensuring connectivity in terms of embedding cost and number of augmented VLinks, respectively. As envisioned at the beginning of this chapter, *CoViNE-fast* successfully restores almost the full bandwidth for the highest priority traffic in the presence of both single and double SLink failures as shown in Fig. 2.9(a) and Fig. 2.9(b), respectively. However, the successful restoration of the highest priority traffic achieved by *CoViNE-fast* comes at the expense of penalizing the traffic from lower priority classes. The overall decrease in restored bandwidth for *CoViNE-fast* with increasing VN LNR is counter-intuitive. This can be explained by observing the overhead shown in Fig. 2.9(d). As VN LNR increases in Fig. 2.9(d), the number of augmented VLinks and the amount of spare bandwidth decrease, thus offering less bandwidth to restore the same or possibly higher amount of bandwidth lost due to failures. Allocating spare bandwidth in the VLinks of a denser VN could help restore more bandwidth upon failure. However, spare bandwidth allocation with guaranteed connectivity is a separate problem that is out of the scope of this chapter.

Fig. 2.9(a) and Fig. 2.9(b) also demonstrate that restored bandwidths of a specific traffic class achieved by *ViNE-ILP* are always lower than those achieved by both *S-CoViNE-fast* and *D-CoViNE-fast*. This is expected since *ViNE-ILP* does not take any measure to guarantee connectivity in the VN embedding against SLink failures. In addition, restored bandwidth of *ViNE-ILP* with the increase in VN LNR follows an opposite (increasing) trend compared to *CoViNE-fast*. The reasons are twofold. First, a higher LNR induces a higher path diversity in a VN that reduces the chances of VN partitioning in the presence of SLink failures. Second, a denser VN has more options for rerouting traffic in the affected VLinks along alternate paths. Despite the increasing trend, restored bandwidths of *ViNE-ILP* do not exceed those of *CoViNE-fast* even in higher LNR VNs, thanks to the disjointeness constraints of *CoViNE-fast*. Furthermore, restored bandwidths of *ViNE-ILP* are very poor in sparse VNs leaving the VNs vulnerable to failures, whereas *CoViNE-fast* offers much better restorability in such VNs.

2.7 Conclusion

In this chapter, we have studied the **Co***nnectivty-aware* **Vi***rtual* **N***etwork* **E***mbedding* (CoViNE) problem that ensures VN connectivity in the presence of multiple substrate link failures. We have presented an ILP formulation, *CoViNE-opt*, that jointly solves three sub-problems of *CoViNE*, namely, VN augmentation, computation of disjointness constraints, and VN embedding. To address the intractability of *CoViNE-opt*, we have separated VN augmentation and disjointness constraint computation from embedding and addressed them sequentially. We have introduced the concept of conflicting set of a VN to efficiently compute disjointness constraints without relying on the exponential number of edge-cuts in a VN as required by *CoViNE-opt*. Given the NP-complete nature of each of the sub-problems and their inter-dependency, we have presented a heuristic algorithm that solves both VN augmentation and conflicting set computation simultaneously. The conflicting set as well as the augmented VN, both produced by the heuristic algorithm, are then used to impose polynomial number of disjointness constraints to the sub-problem of VN embedding. Based on how we address the constrained VN embedding sub-problem, we have provided two more solutions to *CoViNE*, namely *CoViNE-ILP* and *CoViNE-fast*. *CoViNE-ILP* extends a *Multi-commodity Unsplittable Flow* based ILP formulation to address the constrained VN embedding, while *CoViNE-fast* uses a heuristic algorithm to do the same. In contrast to the state-of-the-art, our solutions are generalized to handle multiple substrate link failures for arbitrary topologies.

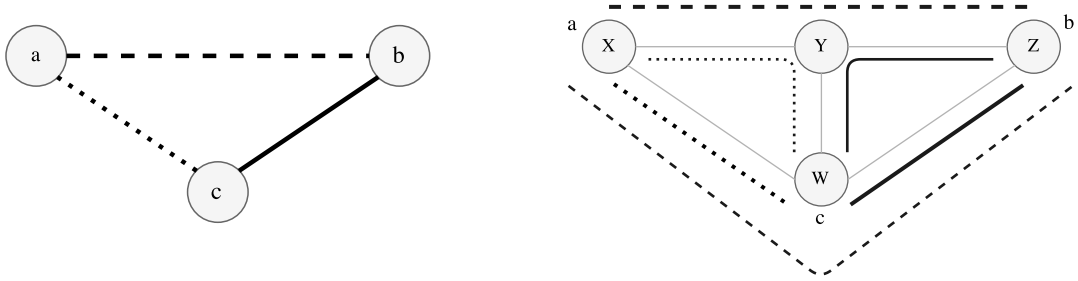
We have evaluated our solutions using a variety of network topologies under different failure scenarios. Evaluation results reveal that although *CoViNE-opt* can be used to benchmark the solutions to CoViNE, it cannot be used to solve practical problems due to its severely low scalability. Our evaluation results demonstrate that *CoViNE-ILP* very closely approximates *CoViNE-opt*, and can be used as a baseline to compare heuristic algorithms. In contrast, *CoViNE-fast* scales to large topologies at the cost of provisioning about 16% additional resources compared to *CoViNE-ILP*, while executing several orders of magnitude faster for the same problem instances. Evaluation results also show that *CoViNE* for single and double link failure scenarios require on average $\sim 24\%$ and $\sim 66\%$ extra resources than a VN embedding strategy that does not guarantee any connectivity. We have also demonstrated that VN connectivity can be leveraged to restore higher priority traffic in the presence of multiple substrate link failures.

Chapter 3

Joint Spare Capacity Allocation and Virtual Network Embedding

The SVNE research literature focuses primarily on protection (*i.e.*, pro-actively provisioning backup during embedding) and restoration (*i.e.*, reactively take action after failure) methods [78]. In this chapter, we focus on the former, *i.e.*, protection mechanism for SVNE, which is usually faster than restoration approaches [59]. When a VN is embedded with SN layer protection, it is the InP's responsibility to handle substrate resource failures. As an illustrative example, consider the embedding of VN abc on SN $WXYZ$ with SN layer survivability in Fig. 3.1, where a, b, and c are virtual nodes and W, X, Y, and Z are substrate nodes. In Fig. 3.1, if the substrate link (X, Y) fails, the InP needs to reroute the affected traffic on the virtual link (a, b) to its backup embedding in the SN, i.e., $\{(X, W), (W, Z)\}$. A key challenge in providing protection is to efficiently utilize resources, since backup resources remain idle until a failure has occurred. Protection based SVNE approaches have adopted different techniques to increase resource efficiency including dedicated backup capacity allocation [173, 48], backup resource sharing [71, 43, 20], multi-path embedding [122, 93], providing weaker forms of survivability [141], and so on.

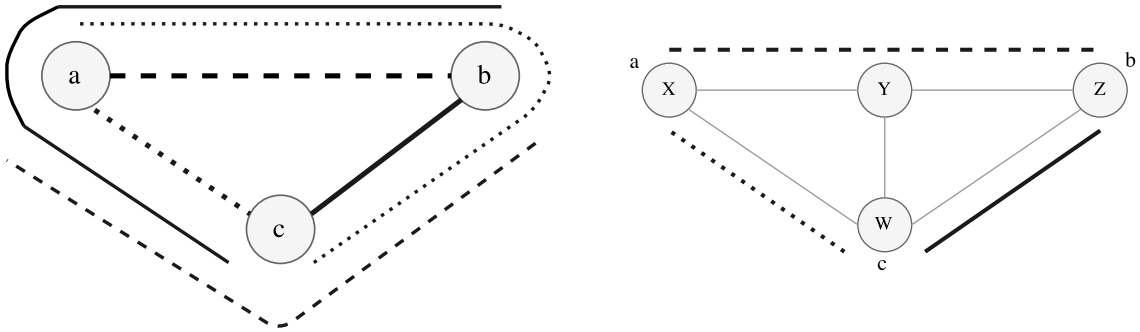
A rather unexplored spectrum in SVNE is to provide protection at the VN layer, much like providing survivability at the upper layer (*e.g.*, Internet Protocol (IP)) of a multi-layer network (*e.g.*, IP over optical such as IP-over-Wavelength Division Multiplexing (WDM) network) [92, 115, 102]. Fig. 3.2 shows a VN embedding with protection at the VN layer. With this type of protection, InPs can offload some of the failure management tasks to SPs by augmenting VNs with sufficient spare capacity for backup and embedding the VNs in a way that primary and backup VN resources are not affected by the same substrate resource failure. When a substrate resource fails, it is the SP's responsibility to reroute



(a) VN topology

(b) VNE with protection in SN

Figure 3.1: Survivability at SN ($WXYZ$) layer. Primary (or backup) embedding of a virtual link is shown with thick (thin) lines. For example, virtual link (a, b) has a primary and backup embedding of $\{(X, Y), (Y, Z)\}$ and $\{(X, W), (W, Z)\}$, respectively.



(a) VN with protection

(b) VN Embedding on SN ($WXYZ$)

Figure 3.2: Survivability at VN (abc) layer. Backup path of a virtual link in a VN is shown with a thin line having same dash pattern. For example, virtual link (a, b) has a backup virtual path $\{(a, c), (c, b)\}$ and an embedding $\{(X, Y), (Y, Z)\}$.

the affected traffic to the pre-allocated backup resources within the VN. For example, in Fig. 3.2, when virtual link (a, b) is affected by the failure of substrate link (X, Y), the SP reroutes traffic between a and b along the backup virtual path $\{(a, c), (b, c)\}$.

The motivation for providing survivability at the VN level is to shift control to SPs, as anticipated in future Transport Software Defined Networks (T-SDNs) [165]. T-SDNs are the next generation of transport networks that leverage Software Defined Networking (SDN) technologies and provide full fledged Virtual Transport Networks (VTNs) instead of traditional end-to-end connectivity to SPs [6]. In essence, an SP can manage its VTN

in the same way as managing its own transport network, and can deploy its own routing, traffic engineering, and failure management solutions [165]. Consequently, with VN layer survivability, the SP can offer different classes of service with differing survivability guarantees to its customers [62]. For instance, the spare backup capacity allocated to the VN of Fig. 3.2(a) can also be used for carrying traffic of a service from best-effort protection class during normal operation, while the same capacity can be used to guarantee protection for a high priority service when substrate resources fail. Such different classes of service and utilization of resources could not be easily achieved in a VN with SN level survivability, since the spare capacity at the SN layer is usually transparent to the SP [52].

Another benefit of providing survivability at the VN layer is that it yields more resource efficient embedding compared to providing the same level of survivability at the SN layer [102]. The intuition behind such claim is that the former can offer more opportunities for spare capacity sharing than the latter. In the case of SN layer protection in Fig. 3.1(b), only the backup capacity on the substrate link (Y, W) can be shared among the backup paths $\{(X, Y), (Y, W)\}$ and $\{(Z, Y), (Y, W)\}$ due to higher path diversity. However, VN layer protection in Fig. 3.2(a) can exploit lower path diversity in a VN to share spare capacities allocated on all the virtual links with one another. Although backup capacity sharing across different VNs could increase the amount of sharing, such sharing is restricted by the fact that virtual links from different VNs may be embedded on different substrate paths to satisfy the location constraints of virtual nodes. These substrate paths may have small number of substrate links in common, resulting in less opportunities of spare capacity sharing than the VN level sharing. We also validate this claim later in our evaluation (§ 3.5). Despite the backup sharing advantages, one can argue that the increased number of signaling and rerouting operations required for VN level survivability may lead to slower restoration compared to ensuring survivability at the SN layer. However, A recent study empirically evaluated the impact of providing protection at the SN level, compared to that at the VN level, on a real testbed [165]. The study shows that: (i) both approaches have similar protection switching time during a failure, and (ii) the latter can accommodate more VNs than the former, thanks to its resource efficiency. InPs can thus increase revenue by adopting VN level survivability without severely impacting failure response times.

A major challenge in SVNE with VN level protection is to jointly optimize spare backup capacity allocation in the VN and survivable VN embedding on the SN. Spare capacity allocation and survivable VN embedding have been studied extensively as independent problems [114, 92, 78] and have been proven to be $\mathcal{NP}$-complete and $\mathcal{NP}$-hard, respectively. SVNE with VN level protection stresses the need to solve these two problems simultaneously, since they can affect each other. For example, an optimal spare capacity allocation without considering VN embedding can be suboptimal, or may even render

the embedding infeasible. Similarly, an optimal VN embedding without consideration for spare capacity allocation may lead to suboptimal or even infeasible spare capacity allocation later. The intricacy of the problem is exacerbated by spare capacity sharing, which is an efficient technique for minimizing aggregate spare capacity [71, 43, 20]. Spare capacity sharing is possible as long as the virtual links, that use the spare capacity as backup in the event of a substrate link failure, are not impacted by the same substrate failure. This requirement can impose constraints such as having a certain number of disjoint paths for backup provisioning that, in turn, can lead to increased embedding cost. Such path disjointedness requirements can be alleviated by allocating dedicated spare capacities instead of sharing the spare capacity. This can also increase embedding cost. Therefore, there is a trade-off between backup path selection and spare backup capacity sharing in a VN and embedding of the VN with disjoint path constraints. Striking a good balance between these choices is challenging, and mandates a thorough investigation. Existing schemes either do not consider all the subproblems of this joint optimization, or address them in separate independent steps, leading to suboptimal solutions [92, 115, 108, 165, 141].

In this chapter, we study the joint optimization problem discussed above with the objective of guaranteeing VN survivability under multiple substrate link failures and minimizing resource usage in the SN. We start with single link failure scenario and then extend the solution to survive multiple link failures. Specifically, we make the following contributions:

- We formulate a joint optimization model using a Quadratic Integer Program (QIP) to optimally solve spare capacity allocation and survivable VN embedding simultaneously. We transform the QIP into an Integer Linear Program (ILP) without sacrificing its optimality. We provide ILP formulations for solving two extreme cases of spare capacity sharing, one of which defines the upper bound of the cost function. We present a mathematical analysis that dictates how the topological properties of the SN affect the level of spare capacity sharing.

- The ILP formulations for the joint optimization problem are not scalable to large problem instances. Hence, we devise an efficient heuristic algorithm to tackle the computational complexity of the ILP-based solutions.

- We perform simulations to evaluate our solutions for single and double link failures. We restrict our evaluation to one and two link failures since the probability of more than two simultaneous link failures is extremely low [65, 117]. We also compare shared backup protection at the VN level with the same at the SN layer proposed in [43].

- We discuss the signaling mechanism and the enabling technology to realize the protection at the VN layer. We also discuss the flexibilities an SP can have by leveraging the spare capacity on the virtual links of a VN.

The rest of this chapter is organized as follows. We present the related literature in § 3.1 and contrast our work with the state-of-the-art. In § 3.2, we present the system model and problem statement followed by a discussion on how spare capacity can be allocated along a virtual link. Then, we present our QIP formulation for the joint optimization problem and the ILP formulations for the optimal and special cases of the problem along with a mathematical analysis in § 3.3. We present the design of our heuristic in § 3.4. The evaluation of our solutions are presented in § 3.5. Finally, we conclude the chapter in § 3.6.

3.1 Related Work

We discuss the state-of-the-art in SVNE with Protection at SN level and VN level in § 3.1.1 and § 3.1.2, respectively. We then contrast our approach with similar works from multi-layer (*e.g.*, IP-over-WDM) network survivability literature in § 3.1.3.

3.1.1 SVNE with Protection at SN

Rahman *et al.*, were the first to address the SVNE problem using a mixed ILP formulation [132]. Since then a number of subsequent research works have addressed different aspects of SVNE such as substrate node failure [176, 177, 171], leveraging multi-path embedding [122, 93], shared backup protection [71, 43, 20], reactive recovery [112, 128, 140, 18], and dedicated VN topology protection [173, 48] among others. Reactive recovery approaches are fundamentally different from our approach. They do not preallocate backup resources and take action after a failure. Therefore, we exclude the reactive approaches from our discussion here.

The SVNE literature exhibits a wide spectrum of protection approaches to improve resource utilization. For instance, shared backup approaches proposed in [71, 43, 20] promote sharing of backup substrate resources for different virtual entities. In contrast, the proposals in [173, 48] go to another extreme and propose to provide dedicated protection for the whole VN topology, *i.e.*, provision a full copy of the VN as a backup. Their motivation is to strictly satisfy failure recovery SLAs in transport networks carrying high volume of traffic. The approaches described in [122, 93] try to optimize backup resource allocation

by assuming in-network multi-path routing support. They do not share backup among the virtual entities, nor do they provide dedicated protection to the VN topology. These proposals assume a virtual link demand can be realized by multiple paths in the SN. Given that the probability of all the embedding paths failing simultaneously is very low, only a fraction of the virtual link demand needs to be provisioned disjointedly as a backup. All the discussed approaches address the SVNE problem from an InP's perspective, *i.e.*, the InP provisions backup resources for a VN in the SN and manages failure recovery tasks. However, they do not explore the solution space where a VN is embedded in a way that an SP can perform failure handling as we study in this chapter.

3.1.2 SVNE with Protection at VN

Compared to the approaches that provide protection at the SN level, only a few SVNE approaches focus on providing protection at the VN level. Among them, an empirical study by Wang *et al.*, [165] compared different protection schemes for VNs in T-SDNs. The results in [165] show that providing protection at the VN level can increase VN acceptance ratio. However, [165] does not provide any mechanism to jointly optimize spare capacity allocation in a VN and embed the VN accordingly. We previously studied a weaker version of the SVNE problem with VN level protection in [141]. This work proposed to embed a VN on an SN in such way that VN connectivity is ensured against multiple substrate link failures. When failure occurs, the SP has to compute alternate paths in the VN to restore the affected traffic which may incur delay. In addition, this work does not guarantee any bandwidth in case of a failure and only allows the VN to operate in a best effort manner. Barla *et al.*, proposed design models for cloud services that provide resiliency either at the VN or at the SN layer [23]. Their resiliency model employs dedicated backup consisting of additional virtual links to reach the recovery data center. In contrast, our approach alleviates the need for changing the VN topology and uses shared spare capacity allocated to existing virtual links to survive link failures.

3.1.3 Network Survivability in Multi-layer Networks

A similar problem to our joint optimization has been studied in the IP-over-WDM literature, namely, *Strongly Survivable Routing (SSR)* [108]. SSR performs mapping and capacity assignment of IP links over lighpaths in a WDM network in way that guarantees connectivity and spare capacity at the IP layer during a failure in either IP or WDM layer. A simplified version of SSR, namely, *Weakly Survivable Routing (WSR)* determines the

mapping of an IP network that remains connected upon a WDM link failure. Although SSR and WSR delegate survivability to the IP layer of a multi-layer network, they do not pre-compute backup paths as we propose in this chapter. After a link fails, they require a time-consuming step for finding alternate paths using the spare capacity in the IP layer. In our approach for VN layer survivability, we obviate such restoration step by computing and storing the backup paths during VN embedding. Despite the difference, we now discuss some prominent works that address SSR as well as spare capacity allocation problem in multi-layer networks.

The authors in [108] and [92] addressed the SSR problem in two separate steps. In the first step, they solve the WSR problem with distinct objectives. Specifically, Lin *et al.*, [108] map IP links on WDM lightpaths such that the smallest capacity of the WDM link in the lightpaths is maximized, whereas Kan *et al.*, [92] minimize the maximum IP bandwidth lost due to a WDM link failure. In the second step, they assign spare capacity to the resultant mapping to ensure that bandwidth demands of the IP links can be rerouted with full capacity after a WDM link fails. However, their two stage approach to the problem may not lead to the optimal solution. More importantly, the first stage of their approach can generate a mapping that may become infeasible in the second stage if the required spare capacity cannot be assigned due to some resource constraints. In addition, they do not consider spare capacity sharing to minimize resource usage.

To improve resource efficiency, Liu *et al.*, [115] leveraged backup capacity sharing by allocating spare capacity in the top layer of a two-layer network. However, this work assumes that the mapping between top and bottom layers is pre-computed and given as input to the spare capacity allocation problem. Hence, this approach suffers from its inability to achieve optimality similar to [108] and [92]. Kubilinskas *et al.*, [102] studied the survivability problem at the IP layer of an IP-over-WDM network using hot standby path protection. Their formulation has two shortcomings. First, it requires a set of pre-computed candidate paths for each IP layer demand. Second, although the formulation supports capacity sharing between a primary path and its standby protection path, it does not support capacity sharing among the protection paths, thus resulting in poor resource utilization. A common feature of the solutions of IP-over-WDM literature is that they assume a fixed placement of IP routers in the network, whereas an SVNE algorithm needs to determine both the mappings of virtual nodes and links. Hence, solutions from IP-over-WDM networks cannot be directly applied to our problem.

3.2 System Model and Background

We first present basic notations in § 3.2.1 and a formal statement of the problem in § 3.2.2. We explain the concept of shared risk groups in § 3.2.3. We then discuss how embedding affects spare capacity allocation on virtual links in § 3.2.4.

3.2.1 Basic Notations

Substrate Network

We represent the substrate network (SN) as an undirected graph, $G = (V, E)$, where V and E denote the set of substrate nodes (SNodes) and links (SLinks), respectively. The set of neighbors of an SNode $u \in V$ is denoted by $\mathcal{N}(u)$. We associate the following attributes with each SLink $(u, v) \in E$: (i) b_{uv} : bandwidth capacity of the SLink (u, v), (ii) C_{uv} : cost of allocating unit bandwidth on (u, v) for a VLink. We assume that the SNodes are network nodes with sufficient capacity to switch traffic at peak rate between any pair of ports. Therefore, we do not consider any node mapping cost or node capacity constraint.

Virtual Network

We represent the virtual network (VN) as an undirected graph $\hat{G} = (\hat{V}, \hat{E})$, where $\hat{V}$ and $\hat{E}$ represent the set of virtual nodes (VNodes) and virtual links (VLinks), respectively. The set of neighbors of a VNode $\hat{v} \in \hat{V}$ is denoted by $\mathcal{N}(\hat{v})$. Each VLink $(\hat{u}, \hat{v}) \in \hat{E}$ has a bandwidth demand $b_{\hat{u}\hat{v}}$. We also have a set of location constraints (LC), $L = \{L(\hat{u}) | L(\hat{u}) \subseteq V, \forall \hat{u} \in \hat{V}\}$, such that a VNode $\hat{u} \in \hat{V}$ can only be provisioned on an SNode $u \in L(\hat{u})$. We use a binary variable $\ell_{\hat{u}u}$ (1 if $\hat{u} \in \hat{V}$ can be provisioned on $u \in V$, 0 otherwise), to represent this location constraint. We denote the spare backup bandwidth allocated to a VLink $(\hat{u}, \hat{v}) \in \hat{E}$ that serves as a backup for other VLinks by $S_{\hat{u}\hat{v}}$. We assume VNs are already K-edge connected to survive K SLink failures. K-edge connectivity is a necessary condition to ensure that at least K edge disjoint backup virtual paths always exist for each VLink $(\hat{u}, \hat{v}) \in \hat{E}$ [108]. However, if a VN has less than K-edge connectivity, any VN augmentation strategy such as [182, 156, 141] can make the VN K-edge connected.

3.2.2 Problem Statement

Given an SN $G = (V, E)$, a VN $\hat{G} = (\hat{V}, \hat{E})$, and LC L:

- For each VLink $(\hat{u}, \hat{v}) \in \hat{E}$, allocate spare capacity along a set of K backup virtual paths (VPaths) $\hat{\mathscr{P}}^K_{\hat{u}\hat{v}} = \{\hat{P}^k_{\hat{u}\hat{v}} | 1 \leq k \leq K\}$ in the VN, where $\hat{P}^k_{\hat{u}\hat{v}}$ is the k_{th} VPath between $\hat{u}$ and $\hat{v}$ such that $\hat{P}^k_{\hat{u}\hat{v}}$ is edge disjoint from $(\hat{u}, \hat{v})$ and from each $\hat{P}^j_{\hat{u}\hat{v}} \in \hat{\mathscr{P}}^K_{\hat{u}\hat{v}}$ with $j \neq k$, and $b_{\hat{u}\hat{v}}$ spare bandwidth is available on the VLinks in $\hat{P}^k_{\hat{u}\hat{v}}$ after $(\hat{u}, \hat{v})$ is affected by an SLink failure.

- Map each VNode $\hat{v} \in \hat{V}$ to exactly one SNode, $u \in V$. Multiple VNodes from the same VN request should not be mapped to the same SNode. However, multiple VNodes from different VNs can share an SNode.

- Map each VLink $(\hat{u}, \hat{v})$ to a non-empty substrate path (SPath) $P_{\hat{u}\hat{v}}$ having sufficient bandwidth to accommodate the primary demand of $(\hat{u}, \hat{v})$ and the spare backup bandwidth allocated on $(\hat{u}, \hat{v})$. A VLink $(\hat{u}, \hat{v})$ and the VLinks on its VPath $\hat{P}^k_{\hat{u}\hat{v}}$ are edge disjointedly mapped on the SN to ensure that SLink failures do not affect them at the same time. Similarly, two VLinks present in the two VPaths, such as $\hat{P}^k_{\hat{u}\hat{v}}$ and $\hat{P}^j_{\hat{u}\hat{v}}$ where $j \neq k$, of the same VLink $(\hat{u}, \hat{v})$ are mapped on edge disjoint SPaths to eliminate the risk of both the VPaths failing together.

- Minimize the total cost of allocating bandwidth on the SN to embed the VN equipped with spare bandwidth.

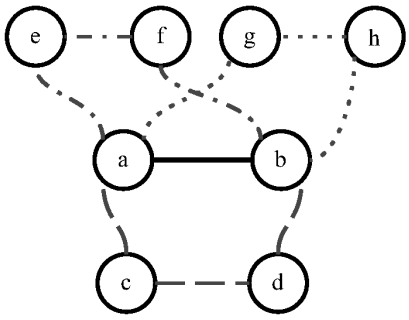
3.2.3 Shared Risk Group

VLinks that share at least one SLink on their mapped SPaths share the same risk since all of them can be impacted if the shared SLink fails. In a context where only single SLink failure is considered, a set of VLinks belong to the same shared risk group (SRG) if and only if they share at least one SLink on their mapped SPaths. In contrast, VLinks that do not share any SLink on their mapped SPaths belong to different SRGs. To represent the SRGs, we partition the VLinks into a number of SRGs represented by the set $D = \{d_1, d_2, d_3, \ldots d_{|D|}\}$, where $|D| \leq |\hat{E}|$. A VLink belongs to exactly one SRG $d_i \in D$ and shares at least one SLink on its mapped SPath with other VLinks in d_i. We use the following variable to decide a VLink's membership to an SRG:

$$d_i^{\hat{u}\hat{v}} = \begin{cases} 1 & \textit{iff } (\hat{u}, \hat{v}) \in \hat{E} \text{ belongs to SRG } d_i \in D, \\ 0 & \text{otherwise.} \end{cases}$$

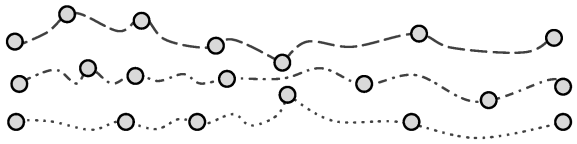
3.2.4 Spare Capacity Assignment Model

Based on how the VLinks form different SRGs during VN embedding, the requirement for spare backup capacity on the VLinks can be different. We explain this fact with a simple example illustrated in Fig. 3.3. In this example, VLink (a, b) is on the backup VPaths of three other VLinks: (c, d), (e, f), and (g, h) as shown in Fig. 3.3(a). We can assign different spare capacity on (a, b) to protect (c, d), (e, f), and (g, h), based on how these three VLinks are mapped. Consider the following scenarios regarding their mappings:

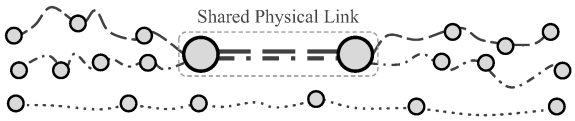


(a) VLink (a, b) is on the backup VPaths of VLinks (c, d), (e, f), (g, h)

(b) (c, d), (e, f), and (g, h) in the same SRG



(c) (c, d), (e, f), and (g, h) in different SRGs



(d) (c, d), (e, f) in one SRG, (g, h) in a different SRG

Figure 3.3: Illustration of different SRGs based on embedding

All three belong to the same SRG. If all three VLinks are in the same SRG, then they share at least one SLink on their mapped SPaths (Fig. 3.3(b)). A single SLink failure can affect all three VLinks. Therefore, spare backup capacity allocated on (a, b) should be sufficient to support the bandwidth requirement of all three VLinks, *i.e.*, $S_{ab} = b_{cd}+b_{ef}+b_{gh}$.

All three belong to different SRGs. If all three VLinks belong to different SRGs, then they do not share any SLink on their mapped SPaths (Fig. 3.3(c)). At most one of the VLinks will be affected by a single SLink failure. Therefore, S_{ab} should be sufficient to support the maximum bandwidth requirement of these three VLinks, *i.e.*, $S_{ab} = \max(b_{cd}, b_{ef}, b_{gh})$.

Two belong to the same SRG, the third in a different SRG. The mapped SPaths can create multiple SRGs out of these three VLinks. For example, in Fig. 3.3(d), VLinks (c, d) and (e, f) belong to the same SRG, whereas VLink (g, h) belongs to a different SRG. A single SLink failure will then affect only one group. Therefore, S_{ab} should be sufficient to support the group with the maximum requirement. For the group with (c, d) and (e, f), the bandwidth requirement is $b_{cd} + b_{ef}$. For the other group, the requirement is b_{gh}. Therefore, spare backup bandwidth on (a, b) should be $\max(b_{cd} + b_{ef}, b_{gh})$.

More formally, if a VLink $(\hat{u}, \hat{v}) \in \hat{E}$ is present on the backup VPaths of a set of VLinks $\hat{\mathcal{H}}_{\hat{u}\hat{v}} \subseteq \hat{E}$, and VLinks in $\hat{E}$ form a set of $D = \{d_1, d_2, d_3, \ldots d_{|D|}\}$ SRGs, we can generalize the spare backup bandwidth allocated to $(\hat{u}, \hat{v})$ as:

$$S_{\hat{u}\hat{v}} = \max_{\forall d_i \in D} \left(\sum_{\forall (\hat{x},\hat{y}) \in \hat{\mathcal{H}}_{\hat{u}\hat{v}}} d_i^{\hat{x}\hat{y}} b_{\hat{x}\hat{y}} \right) \tag{3.1}$$

3.3 Problem Formulation

We first provide a QIP formulation to optimally solve the joint spare capacity allocation and survivable embedding problem for the case of single substrate link failure in § 3.3.1. We then describe the transformation of the QIP to an ILP, namely *Opt-ILP*, in § 3.3.2. However, due to an overwhelming number of decision variables and constraints, *Opt-ILP* is only scalable to very small problem instances. Therefore, in § 3.3.3, we present simplified ILP formulations for two special cases of the joint optimization problem, along with a mathematical analysis that dictates how to select one of the ILP formulations based on the topological properties of an SN. Finally, § 3.3.4 discusses how we can extend our solution to handle multiple independent SLink failures.

3.3.1 Quadratic Integer Program Formulation

We first present our decision variables (§ 3.3.1). Then we introduce the constraints (§ 3.3.1) followed by the objective function of our formulation (§ 3.3.1).

Decision Variables

For each VLink $(\hat{u}, \hat{v}) \in \hat{E}$, there is a backup VPath $\hat{P}_{\hat{u}\hat{v}}$ that provides protection to $(\hat{u}, \hat{v})$ from a single SLink failure. When any SLink on the VLink's mapped SPath fails, $\hat{P}_{\hat{u}\hat{v}}$

provides the full bandwidth $b_{\hat{u}\hat{v}}$ between the VNodes $\hat{u}$ and $\hat{v}$. The following decision variable defines whether a VLink $(\hat{u}, \hat{v}) \in \hat{E}$ belongs to the VPath protecting a VLink $(\hat{x}, \hat{y}) \in \hat{E}$:

$$z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}} = \begin{cases} 1 & \text{if } (\hat{u}, \hat{v}) \in \hat{E} \text{ is on the backup VPath of } (\hat{x}, \hat{y}) \in \hat{E}, \\ 0 & \text{otherwise.} \end{cases}$$

Note that, $z_{\hat{u}\hat{v}}^{\hat{u}\hat{v}} = 0$, since a VLink's backup VPath has to be edge disjoint from itself.

The following decision variable indicates the mapping between a VLink $(\hat{u}, \hat{v}) \in \hat{E}$ and an SLink $(u, v) \in E$:

$$x_{uv}^{\hat{u}\hat{v}} = \begin{cases} 1 & \text{if } (\hat{u}, \hat{v}) \in \hat{E} \text{ is mapped to } (u, v) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

The VNode to SNode mapping is denoted using the following decision variable:

$$y_{\hat{u}u} = \begin{cases} 1 & \text{if } \hat{u} \in \hat{V} \text{ is mapped to } u \in V, \\ 0 & \text{otherwise.} \end{cases}$$

As discussed in § 3.2.3, VLinks that share at least one SLink on their mapped SPaths belong to the same SRG. SRG membership is defined using the decision variable $d_i^{\hat{u}\hat{v}}$, presented in § 3.2.3.

Constraints

VNode Mapping Constraints (3.2) and (3.3) ensure that each VNode of a VN is provisioned on an SNode satisfying the provided location constraints. Moreover, (3.4) constraints an SNode to host at most one VNode from the same VN. Note that VNode mapping follows from the VLink mapping, since there is no cost associated with the VNode mapping.

$$\forall \hat{u} \in \hat{V}, \forall u \in V : y_{\hat{u}u} \leq \ell_{\hat{u}u} \tag{3.2}$$

$$\forall \hat{u} \in \hat{V} : \sum_{u \in V} y_{\hat{u}u} = 1 \tag{3.3}$$

$$\forall u \in V : \sum_{\hat{u} \in \hat{V}} y_{\hat{u}u} \leq 1 \tag{3.4}$$

Backup VPath Continuity Constraints A VLink in a VN is protected by a VPath in the VN to survive a single SLink failure. (3.5) ensures continuity of a backup VPath protecting a VLink $(\hat{x}, \hat{y}) \in \hat{E}$:

$$\forall (\hat{x}, \hat{y}) \in \hat{E} : \sum_{\forall \hat{v} \in \mathcal{N}(\hat{u}) \setminus \{\hat{y}\}} (z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}} - z_{\hat{x}\hat{y}}^{\hat{v}\hat{u}}) = \begin{cases} 1 & \text{if } \hat{u} = \hat{x} \\ -1 & \text{if } \hat{u} = \hat{y} \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

VLink Mapping Constraints We ensure that every VLink is mapped to a non-empty set of SLinks using (3.6). Then, (3.7) makes sure that the in-flow and out-flow of each SNode is equal, except for the SNodes where the endpoints of a VLink are mapped. This ensures that the non-empty set of SLinks corresponding to a VLink's mapping form a single continuous SPath.

$$\forall (\hat{u}, \hat{v}) \in \hat{E} : \sum_{\forall (u,v) \in E} x_{uv}^{\hat{u}\hat{v}} \geq 1 \tag{3.6}$$

$$\forall \hat{u}, \hat{v} \in \hat{V}, \forall u \in V : \sum_{\forall v \in \mathcal{N}(u)} (x_{uv}^{\hat{u}\hat{v}} - x_{vu}^{\hat{u}\hat{v}}) = y_{\hat{u}u} - y_{\hat{v}u} \tag{3.7}$$

The binary nature of the VLink mapping decision variable and the flow constraint prevent any VLink from being mapped to more than one SPaths, thus, restricting the VLink mapping to the *Multi-commodity Unsplittable Flow Problem* [58].

Capacity Constraints We also need to ensure that we do not over-commit the bandwidth resources we have on the SLinks. To do so, we first compute the spare backup bandwidth allocated to a VLink $(\hat{u}, \hat{v}) \in \hat{E}$ using (3.1) as follows:

$$S_{\hat{u}\hat{v}} = \max_{\forall d_i \in D} \sum_{\forall (\hat{x},\hat{y}) \in \hat{E} \setminus \{(\hat{u},\hat{v})\}} z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}} \times d_i^{\hat{x}\hat{y}} \times b_{\hat{x}\hat{y}} \tag{3.8}$$

Then, the following constraints prevent any over-commit of the bandwidth resource on the SLinks:

$$\forall (u, v) \in E : \sum_{\forall (\hat{u},\hat{v}) \in \hat{E}} x_{uv}^{\hat{u}\hat{v}} \times (b_{\hat{u}\hat{v}} + S_{\hat{u}\hat{v}}) \leq b_{uv} \tag{3.9}$$

Note that (3.9) is a cubic constraint, since $S_{\hat{u}\hat{v}}$ is quadratic according to (3.8). Therefore, we take the following steps to linearize $S_{\hat{u}\hat{v}}$ in order to keep (3.9) in quadratic order. First, we introduce a new variable $g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i)$, defined as follows:

$$g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i) = \begin{cases} 0 & \text{if } z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}} = 1 \text{ and } d_i^{\hat{x}\hat{y}} = 1, \\ 1 & \text{otherwise.} \end{cases}$$

Essentially, for a given VLink $(\hat{u}, \hat{v}) \in \hat{E}$, the zero values of $g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i)$ induce a set of VLinks that belong to the same SRG and have $(\hat{u}, \hat{v})$ on their backup VPaths. The value of $g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i)$ is set using the following constraint:

$$\forall (\hat{u}, \hat{v}) \in \hat{E}, \forall (\hat{x}, \hat{y}) \in \hat{E}, \forall d_i \in D : z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}} + d_i^{\hat{x}\hat{y}} + g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i) \leq 2 \tag{3.10}$$

We can use $g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i)$ to rewrite (3.8) in a linear form as follows:

$$S_{\hat{u}\hat{v}} = \max_{\forall d_i \in D} \sum_{\forall (\hat{x},\hat{y}) \in \hat{E} \setminus \{(\hat{u},\hat{v})\}} (1 - g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i)) \times b_{\hat{x}\hat{y}} \tag{3.11}$$

Since our objective function will be a minimization function, we define $g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i)$ such that setting it to 1 minimizes the value of $S_{\hat{u}\hat{v}}$, unless it is constrained to be 0 according to (3.10). This constrained case will only occur when both $z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}$ and $d_i^{\hat{x}\hat{y}}$ are 1, as enforced by (3.10).

SRG Constraints The mapped SPaths of the VLinks from an SRG d_i, must be edge disjoint from the mapped SPaths of the VLinks from a different SRG d_j ($\forall j \neq i$). This is accounted for in (3.12). (3.13) ensures that two VLinks from the same SRG share at least one SLink on their mapped SPaths. Note that a VLink $(\hat{u}, \hat{v}) \in \hat{E}$ cannot be present in more than one SRGs, which is satisfied by (3.14).

$$\begin{aligned} &\forall (u,v) \in E, \forall (\hat{u},\hat{v}) \in \hat{E}, \forall (\hat{x},\hat{y}) \in \hat{E}, \\ &\forall d_i \in D, \forall d_j \in D \ s.t. \ (\hat{u},\hat{v}) \neq (\hat{x},\hat{y}) \text{ and } i \neq j : \\ &d_i^{\hat{u}\hat{v}} + d_j^{\hat{x}\hat{y}} + x_{uv}^{\hat{u}\hat{v}} + x_{vu}^{\hat{u}\hat{v}} + x_{uv}^{\hat{x}\hat{y}} + x_{vu}^{\hat{x}\hat{y}} \leq 3 \end{aligned} \tag{3.12}$$

$$\begin{aligned} &\forall d_i \in D, \forall (\hat{u},\hat{v}) \in \hat{E}, \forall (\hat{x},\hat{y}) \in \hat{E} \ s.t. \ (\hat{u},\hat{v}) \neq (\hat{x},\hat{y}), \\ &\exists (u,v) \in E : d_i^{\hat{u}\hat{v}} + d_i^{\hat{x}\hat{y}} + x_{uv}^{\hat{u}\hat{v}} + x_{vu}^{\hat{u}\hat{v}} + x_{uv}^{\hat{x}\hat{y}} + x_{vu}^{\hat{x}\hat{y}} = 4 \end{aligned} \tag{3.13}$$

$$\forall (\hat{x},\hat{y}) \in \hat{E} : \sum_{\forall d_i \in D} d_i^{\hat{x}\hat{y}} = 1 \tag{3.14}$$

Survivability Constraints To ensure survivability of the VN under single SLink failure, the mapped SPath of a VLink cannot share any SLink with the mapped SPaths of the VLinks present on its backup VPath. The following constraints make sure this edge disjointedness requirement:

$$\begin{aligned} &\forall (u,v) \in E, \forall ((\hat{u},\hat{v}),(\hat{x},\hat{y})) \in \hat{E} \times \hat{E} \ s.t. \ (\hat{u},\hat{v}) \neq (\hat{x},\hat{y}) : \\ &z_{\hat{u}\hat{v}}^{\hat{x}\hat{y}} + x_{uv}^{\hat{u}\hat{v}} + x_{vu}^{\hat{u}\hat{v}} + x_{uv}^{\hat{x}\hat{y}} + x_{vu}^{\hat{x}\hat{y}} \leq 2 \end{aligned} \tag{3.15}$$

Objective Function

As per the problem statement presented in § 3.2.2, we do not consider any node mapping cost in our VN embedding. Thus, our cost function minimizes the total cost of provisioning the working and spare backup bandwidth for the VLinks of a VN on the SLinks of an SN. This gives us the following objective function:

$$\text{minimize} \left(\sum_{\forall (\hat{u},\hat{v}) \in \hat{E}} \sum_{\forall (u,v) \in E} x_{uv}^{\hat{u}\hat{v}} \times C_{uv} \times (b_{\hat{u}\hat{v}} + S_{\hat{u}\hat{v}}) \right) \tag{3.16}$$

3.3.2 ILP Transformation, *Opt-ILP*

Our formulation for the joint optimization problem has a quadratic constraint (3.9) and a quadratic objective function (3.16). Therefore, the QIP presented in § 3.3.1 is a Quadratically Constrained Quadratic Program (QCQP) and falls into the general category of the Quadratic Assignment Problem (QAP) [106]. Solving a QAP is computationally expensive and is known to be $\mathcal{NP}$-hard [36]. Sahni *et al.*, proved that even finding an $\epsilon-$approximate solution of QAP is $\mathcal{NP}$-hard [139]. We now present the steps to linearize the QIP by using a technique similar to the one discussed in [124]. For the purpose of linearization, we first put a bound on the spare backup bandwidth of a VLink $(\hat{u}, \hat{v}) \in \hat{E}$, *i.e.*, $S_{\hat{u}\hat{v}}$: $0 \leq S_{\hat{u}\hat{v}} \leq \lambda$, where λ is a very large value. We also introduce a new integer variable $q_{uv}^{\hat{u}\hat{v}}$, defined in terms of $x_{uv}^{\hat{u}\hat{v}}$ as follows:

$$q_{uv}^{\hat{u}\hat{v}} = \begin{cases} S_{\hat{u}\hat{v}} & \text{if } x_{uv}^{\hat{u}\hat{v}} = 1, \\ 0 & \text{if } x_{uv}^{\hat{u}\hat{v}} = 0. \end{cases}$$

The following constraints enforce the above definition.

$$\forall (u,v) \in E, \forall (\hat{u},\hat{v}) \in \hat{E} : q_{uv}^{\hat{u}\hat{v}} \geq 0 \tag{3.17}$$

$$\forall (u,v) \in E, \forall (\hat{u},\hat{v}) \in \hat{E} : S_{\hat{u}\hat{v}} - \lambda \times (1 - x_{uv}^{\hat{u}\hat{v}}) \leq q_{uv}^{\hat{u}\hat{v}} \tag{3.18}$$

To elaborate, when $x_{uv}^{\hat{u}\hat{v}} = 0$, constraints (3.17) and (3.18) become $q_{uv}^{\hat{u}\hat{v}} \geq 0$ and $S_{\hat{u}\hat{v}} - \lambda \leq q_{uv}^{\hat{u}\hat{v}}$, respectively. Since λ is a very large value by definition, the constraints finally reduce to $q_{uv}^{\hat{u}\hat{v}} \geq 0$. On the other hand, when $x_{uv}^{\hat{u}\hat{v}} = 1$, constraint (3.17) and (3.18) become $q_{uv}^{\hat{u}\hat{v}} \geq 0$ and $S_{\hat{u}\hat{v}} \leq q_{uv}^{\hat{u}\hat{v}}$, respectively. In this latter case, constraint (3.18), *i.e.*, $S_{\hat{u}\hat{v}} \leq q_{uv}^{\hat{u}\hat{v}}$ dominates. Finally, if we include $q_{uv}^{\hat{u}\hat{v}}$ in the minimization objective function, the smallest possible value of $q_{uv}^{\hat{u}\hat{v}}$ will be used to minimize the value of the objective function, yielding $q_{uv}^{\hat{u}\hat{v}} = S_{\hat{u}\hat{v}}$ (for $x_{uv}^{\hat{u}\hat{v}} = 1$) and $q_{uv}^{\hat{u}\hat{v}} = 0$ (for $x_{uv}^{\hat{u}\hat{v}} = 0$).

We now rewrite the capacity constraint (3.9) as the following linear constraint using $q_{uv}^{\hat{u}\hat{v}}$.

$$\forall (u,v) \in E : \sum_{\forall (\hat{u},\hat{v}) \in \hat{E}} (x_{uv}^{\hat{u}\hat{v}} \times b_{\hat{u}\hat{v}} + q_{uv}^{\hat{u}\hat{v}}) \leq b_{uv} \tag{3.19}$$

Similarly, the quadratic objective function can be written in a linearized form as follows:

$$\text{minimize} \left(\sum_{\forall(\hat{u},\hat{v})\in\hat{E}} \sum_{\forall(u,v)\in E} x_{uv}^{\hat{u}\hat{v}} \times C_{uv} \times b_{\hat{u}\hat{v}} + C_{uv} \times q_{uv}^{\hat{u}\hat{v}}) \right) \tag{3.20}$$

3.3.3 Problem Variations

Opt-ILP is scalable to very small problem instances due to its large number of constraints and variables. For instance, the number of constraints generated by capacity constraints (3.10) and SRG constraints (3.12) and (3.13) are $\hat{E}^3$, $E \times \hat{E}^4$, and $E \times \hat{E}^3$, respectively. Similarly, the number of variables generated by $d_i^{\hat{x}\hat{y}}$ and $g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i)$ are $\hat{E}^2$ and $\hat{E}^3$. To reduce problem complexity, we formulate two simpler variants of *Opt-ILP*, namely *Max-ILP* and *Min-ILP*, that represent maximum and minimum sharing of spare capacity, respectively.

ILP for Maximum Spare Capacity Sharing, *Max-ILP*

Design of *Max-ILP* is motivated by an observation from our evaluation results that for a VLink $(\hat{u},\hat{v})$, *Opt-ILP* assigns the VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$ into separate SRGs whenever VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$ can be mapped to disjoint SPaths, thus preferring more sharing of the spare capacity. Therefore, *Max-ILP* enforces maximum sharing of spare backup capacity $S_{\hat{u}\hat{v}}$ of a VLink $(\hat{u},\hat{v})$ among the VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$ that have $(\hat{u},\hat{v})$ in their VPaths, similar to the example shown in Fig. 3.3(c). In order to guarantee full bandwidth between pairs of VNodes during an SLink failure, *Max-ILP* forcefully assigns VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$ into separate SRGs instead of deciding the assignment dynamically during embedding. In this way, *Max-ILP* eliminates decision variables $d_i^{\hat{x}\hat{y}}$ and $g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i)$ from the capacity and SRG constraints of *Opt-ILP*. Therefore, in *Max-ILP*, we can rewrite (3.11) as follows and eliminate the constraints in (3.10):

$$S_{\hat{u}\hat{v}} = \max_{\forall(\hat{x},\hat{y})\in\hat{E}\setminus\{(\hat{u},\hat{v})\}} z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}} \times b_{\hat{x}\hat{y}} \tag{3.21}$$

Similarly, we can replace SRG constraints (3.12), (3.13), and (3.14) of *Opt-ILP* by the following disjointedness constraints:

$$\forall(u,v) \in E, \forall(\hat{u},\hat{v}) \in \hat{E}, \forall(\hat{x},\hat{y}) \in \hat{E} \setminus \{(\hat{u},\hat{v})\},$$
$$\forall(\hat{a},\hat{b}) \in \hat{E} \setminus \{(\hat{u},\hat{v}),(\hat{x},\hat{y})\} : \tag{3.22}$$
$$z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}} + z_{\hat{a}\hat{b}}^{\hat{u}\hat{v}} + x_{uv}^{\hat{x}\hat{y}} + x_{vu}^{\hat{x}\hat{y}} + x_{uv}^{\hat{a}\hat{b}} + x_{vu}^{\hat{a}\hat{b}} \leq 3$$

These disjointedness constraints ensure that if a VLink $(\hat{u},\hat{v}) \in \hat{E}$ is in the backup VPaths of two other VLinks such as $(\hat{x},\hat{y}) \in \hat{E} \setminus (\hat{u},\hat{v})$ and $(\hat{a},\hat{b}) \in \hat{E} \setminus (\hat{u},\hat{v})$, $(\hat{x},\hat{y})$ and $(\hat{a},\hat{b})$ cannot share an SLink in their mappings. However, $(\hat{x},\hat{y})$ and $(\hat{a},\hat{b})$ can share an SLink if their VPaths do not have any common VLink. Therefore, *Max-ILP* consists of the constraints (3.21), (3.22), and all the constraints of *Opt-ILP* except the constraints (3.10), (3.11), (3.12), (3.13), and (3.14). The same objective function (3.20) of *Opt-ILP* prevails in *Max-ILP*. Thus, *Max-ILP* replaces higher order variables and constraints of *Opt-ILP* by a lower order constraint. The disadvantage of *Max-ILP* is that it may require more disjoint SPaths to satisfy constraints of (3.22) than those required by *Opt-ILP* .

ILP for No Spare Capacity Sharing, *Min-ILP*

Max-ILP requires that all the VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$ are mapped to disjoint SPaths. This is only achievable in SNs where adequate disjoint SPaths can be found. However, in sparse SNs, *Max-ILP* may become infeasible due to the unavailability of disjoint SPaths. Such infeasibility can also occur in SNs having *trap topological* structure [55, 69]. We have observed this behavior in our evaluation for embedding VNs in sparser SNs. Despite the infeasibility of *Max-ILP*, *Opt-ILP* is able to find a solution in such SNs by striking a balance between the level of spare capacity sharing and the number of disjoint SPaths. Following this observation, we present an ILP formulation, *Min-ILP*, that does not allow any sharing of spare backup capacity as shown in Fig. 3.3(b). Consequently, all the VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$ belong to only one SRG and the SRG constraints (3.12), (3.13), and (3.14) of *Opt-ILP* can be eliminated from *Min-ILP*. In addition, we can exclude decision variables $d_i^{\hat{x}\hat{y}}$ and $g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i)$ and the constraint (3.11) from *Min-ILP*. However, the disadvantage of *Min-ILP* is that all the VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$ can be affected during an SLink failure due to sharing of the same risk. Hence, the spare backup capacity of a VLink $S_{\hat{u}\hat{v}}$ in *Min-ILP* should be adequate enough to serve the bandwidth of all the VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$. This increases the spare backup capacity requirement $S_{\hat{u}\hat{v}}$ of a VLink as shown by the following equation:

$$S_{\hat{u}\hat{v}} = \sum_{\forall(\hat{x},\hat{y}) \in \hat{E} \setminus \{(\hat{u},\hat{v})\}} z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}} \times b_{\hat{x}\hat{y}} \tag{3.23}$$

Similar to *Max-ILP*, *Min-ILP* consists of the constraints (3.23) and all the constraints of *Opt-ILP* except the constraints (3.10), (3.11), (3.12), (3.13), and (3.14). The same objective function (3.20) of *Opt-ILP* prevails in *Min-ILP* as well. In this way, *Min-ILP* replaces all the higher order decision variables and constraints of *Opt-ILP*. Another potential drawback of *Min-ILP* is that it has to explore a larger search space than *Max-ILP* would require due to *Min-ILP*'s lower number of constraints.

Comparative Study between *Max-ILP* and *Min-ILP*

Recall from § 3.2.4 that based on how the VLinks are sharing risks in their mappings, the spare capacity allocation can be different on a VLink. One extreme case is when all the VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$ are mapped disjointedly (*Max-ILP*). Another extreme case is when there is minimal disjointedness in the mapping, *i.e.*, VLinks are mapped disjointedly only when constrained by the backup VPath's survivability constraints (3.15) (*Min-ILP*). We now present a mathematical analysis showing how the mapped SPath length affects the preference for disjointedness.

Let us assume for the sake of simplicity that VLinks in $\hat{H}_{\hat{u}\hat{v}}$ are not on the backup VPath of any other VLink. We represent the mean mapped SPath length for *Max-ILP* and *Min-ILP* as $\mathcal{P}$ and $\mathcal{R}$, respectively. Note that $\mathcal{P}$ and $\mathcal{R}$ should be different since *Max-ILP* has to satisfy disjointedness constraints (3.22), thus resulting in longer SPaths. On the other hand, SPaths in *Min-ILP* have no such disjointedness requirements to adhere to, thereby yielding shorter SPaths. Finally, we assume a unit cost of allocating bandwidth on the SLink, *i.e.*, $\forall(u,v) \in E : C_{uv} = 1$. In *Max-ILP*, each VLink in $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$ belongs to different SRGs. Using (3.21), spare capacity requirement for *Max-ILP* is:

$$S^{max}_{\hat{u}\hat{v}} = \max_{\forall(\hat{x},\hat{y})\in\hat{\mathcal{H}}_{\hat{u}\hat{v}}} b_{\hat{x}\hat{y}} \tag{3.24}$$

In *Min-ILP*, all the VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$ for a given $(\hat{u},\hat{v}) \in \hat{E}$ belong to the same SRG. Therefore, we can compute spare capacity requirement for *Min-ILP* using (3.23) as follows:

$$S^{min}_{\hat{u}\hat{v}} = \sum_{\forall(\hat{x},\hat{y})\in\hat{\mathcal{H}}_{\hat{u}\hat{v}}} b_{\hat{x}\hat{y}} \tag{3.25}$$

For *Max-ILP*, the cost of mapping the VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}} \cup \{(\hat{u},\hat{v})\}$ is obtained by combining (3.16) and (3.24) as follows:

$$cost^{max} = \mathcal{P} \times \left(b_{\hat{u}\hat{v}} + \max_{\forall(\hat{x},\hat{y})\in\hat{\mathcal{H}}_{\hat{u}\hat{v}}} b_{\hat{x}\hat{y}} + \sum_{\forall(\hat{x},\hat{y})\in\hat{\mathcal{H}}_{\hat{u}\hat{v}}} b_{\hat{x}\hat{y}} \right)$$

Similarly, the cost of mapping the VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}} \cup \{(\hat{u},\hat{v})\}$ for *Min-ILP* can be obtained by combining (3.16) and (3.25) as:

$$cost^{min} = \mathcal{R} \times \left(b_{\hat{u}\hat{v}} + 2 \sum_{\forall(\hat{x},\hat{y})\in\hat{\mathcal{H}}_{\hat{u}\hat{v}}} b_{\hat{x}\hat{y}} \right)$$

Now, $cost^{max} < cost^{min}$ will be true if:

$$\mathcal{P} \times \left(b_{\hat{u}\hat{v}} + \max_{\forall(\hat{x},\hat{y})\in\hat{\mathcal{H}}_{\hat{u}\hat{v}}} b_{\hat{x}\hat{y}} + \sum_{\forall(\hat{x},\hat{y})\in\hat{\mathcal{H}}_{\hat{u}\hat{v}}} b_{\hat{x}\hat{y}} \right) < \mathcal{R} \times \left(b_{\hat{u}\hat{v}} + 2 \sum_{\forall(\hat{x},\hat{y})\in\hat{\mathcal{H}}_{\hat{u}\hat{v}}} b_{\hat{x}\hat{y}} \right)$$

This gives us the following final inequality:

$$\frac{\mathcal{P}}{\mathcal{R}} < \frac{b_{\hat{u}\hat{v}} + 2 \times \sum_{\forall(\hat{x},\hat{y})\in\hat{\mathcal{H}}_{\hat{u}\hat{v}}} b_{\hat{x}\hat{y}}}{b_{\hat{u}\hat{v}} + \max_{\forall(\hat{x},\hat{y})\in\hat{\mathcal{H}}_{\hat{u}\hat{v}}} b_{\hat{x}\hat{y}} + \sum_{\forall(\hat{x},\hat{y})\in\hat{\mathcal{H}}_{\hat{u}\hat{v}}} b_{\hat{x}\hat{y}}}$$

When the set $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$ is sufficiently large, we have $b_{\hat{u}\hat{v}} \leq \max_{\forall(\hat{x},\hat{y})\in\hat{\mathcal{H}}_{\hat{u}\hat{v}}} b_{\hat{x}\hat{y}} << \sum_{\forall(\hat{x},\hat{y})\in\hat{\mathcal{H}}_{\hat{u}\hat{v}}} b_{\hat{x}\hat{y}}$, yielding $P <\sim 2R$. This gives us the following insight: *as long as the mean mapped SPath length of the Max-ILP does not become about twice the mean mapped SPath length of the Min-ILP, Max-ILP yields an embedding with a lower cost than Min-ILP.* In other words, cost of *Max-ILP* will only exceed the cost of *Min-ILP* when the increase in mean SPath

length due to satisfying disjointedness constraints (3.22) is almost equal to the mean SPath length computed without the disjointedness constraints (3.22). Such situation can occur in sparse SNs that lack path diversity. In contrast, dense SNs have higher path diversity and higher number of SLinks. Consequently, the number of edge-disjoint SPaths is also higher in denser SNs than in sparser SNs [31]. To satisfy disjointedness constraints, denser SNs increase mean SPath length by a smaller factor compared to that in sparser SNs. Therefore, *Min-ILP* should be the preferred choice only in extremely sparse SNs, whereas *Max-ILP* can be used in all other types of SNs. This result motivates us to develop a heuristic algorithm that prefers sharing of spare capacity much like the case of *Max-ILP*.

3.3.4 Single to Multiple SLink Failures

We now discuss how our solutions for single SLink failure can be extended to handle multiple SLink failures. Since *Max-ILP* closely approximates *Opt-ILP* (see § 3.5.3) and *Max-ILP* is more resource efficient than *Min-ILP* (see § 3.3.3), we focus on extending *Max-ILP* for multiple SLink failures. As discussed in [76], there are two methods to survive against multiple (say K) link failures in a single layer network, a VN in our case. In the first method, for each VLink $(\hat{u}, \hat{v})$, provision spare capacities across K edge-disjoint backup VPaths so that at least one of them provides the full bandwidth even if $(\hat{u}, \hat{v})$ and $K-1$ of the backup VPaths are affected by K simultaneous VLink failures. During VN embedding, the disjoint path requirement translates to the following: a VLink in the k-th edge-disjoint backup VPath (including $(\hat{u}, \hat{v})$) and another VLink in the j-th ($j \neq k$) edge-disjoint VPath should be embedded on disjoint SPaths. In the second method, for each VLink $(\hat{u}, \hat{v})$, provision only one backup VPath $\hat{P}_{\hat{u}\hat{v}}$ such that the VLinks in $\hat{P}_{\hat{u}\hat{v}}$ have enough spare capacity to carry all the traffic of any K VLinks in the VN. In this case, rerouting of traffic during the failure of $(\hat{u}, \hat{v})$ is the same as previous case. However, during the failures of $(\hat{u}, \hat{v})$ and any VLink $(\hat{x}, \hat{y}) \in \hat{P}_{\hat{u}\hat{v}}$, traffic is rerouted to the VPath consisting of $(\hat{P}_{\hat{u}\hat{v}} - (\hat{x}, \hat{y})) \cup \hat{P}_{\hat{x}\hat{y}}$, where $\hat{P}_{\hat{x}\hat{y}}$ is the backup VPath between $\hat{x}$ and $\hat{y}$. This method requires lesser number of disjoint paths compared to the first method, *i.e.*, only $(\hat{u}, \hat{v})$ and any VLink in $\hat{P}_{\hat{u}\hat{v}}$ need to be embedded on disjoint SPaths. However, spare capacity requirement can be unnecessarily high since a VLink in $\hat{P}_{\hat{x}\hat{y}}$ has to carry the traffic of K other VLinks that may have been impacted by K SLink failures, limiting the applicability of the second method. An additional disadvantage of this method is that rerouted traffic may have to traverse many VLinks chained through backup VPaths when K VLinks fail [76]. Hence, we adopt the first method in extending *Max-ILP* to handle the case of multiple SLink failures as discussed in the following. First, we modify the decision

variable defining VPath relationships to take into account multiple VPaths as below:

$$z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(k) = \begin{cases} 1 & \text{if } (\hat{u},\hat{v}) \text{ is on the } k_{th} \text{ backup VPath of } (\hat{x},\hat{y}), \\ 0 & \text{otherwise.} \end{cases}$$

Similar to (3.5), we need to have VPath continuity constraints for each of backup VPaths of a VLink. In addition, we need the following constraints to ensure the edge-disjointedness of the K backup VPaths of a VLink.

$$\begin{aligned} \forall k = 1,2,...,K, \forall j = 1,2,...,K \ s.t. \ k \neq j, \\ \forall ((\hat{u},\hat{v}),(\hat{x},\hat{y})) \in \hat{E} \times \hat{E} \ s.t. \ (\hat{u},\hat{v}) \neq (\hat{x},\hat{y}) : \\ z_{\hat{u}\hat{v}}^{\hat{x}\hat{y}}(k) + z_{\hat{u}\hat{v}}^{\hat{x}\hat{y}}(j) \leq 1 \end{aligned} \tag{3.26}$$

Following (3.15), we need survivability constraints between the mappings of a VLink and the VLinks in each of its backup VPaths. We also need to ensure that two VLinks present in two backup VPaths of a VLink are embedded on disjoint SPaths:

$$\begin{aligned} \forall k = 1,2,...,K, \forall j = 1,2,...,K \ s.t. \ k \neq j, \\ \forall (\hat{x},\hat{y}), \forall ((\hat{u},\hat{v}),(\hat{a},\hat{b})) \in \hat{E} \times \hat{E} \ s.t. \ (\hat{u},\hat{v}) \neq (\hat{a},\hat{b}) : \\ z_{\hat{u}\hat{v}}^{\hat{x}\hat{y}}(k) + z_{\hat{a}\hat{b}}^{\hat{x}\hat{y}}(j) + x_{uv}^{\hat{u}\hat{v}} + x_{vu}^{\hat{u}\hat{v}} + x_{uv}^{\hat{a}\hat{b}} + x_{vu}^{\hat{a}\hat{b}} \leq 3 \end{aligned} \tag{3.27}$$

Finally, the spare capacity constraints in (3.21) need to be modified to handle K SLink failures. Ideally, the spare capacity $S_{\hat{u}\hat{v}}$ on a VLink $(\hat{u},\hat{v})$ should be sufficient to carry the traffic of any K VLinks that have $(\hat{u},\hat{v})$ in any of their backup VPaths. Accordingly, we modify the spare capacity requirement as follows:

$$S_{\hat{u}\hat{v}} = \max_{\forall \xi \subset \hat{E} \ s.t. \ |\xi| = K} \sum_{\forall (\hat{x},\hat{y}) \in \xi} \sum_{1 \leq k \leq K} z_{\hat{u}\hat{v}}^{\hat{x}\hat{y}}(k) \times b_{\hat{x}\hat{y}} \tag{3.28}$$

As discussed in [76, 77], not all backup VPaths are used simultaneously even for $K = 2$ in a single layer VN. In fact, spare capacity along VPaths can be efficiently shared by introducing constraints similar to the ones presented in [77] and by taking into account different embedding options of the VLinks of a VN. Extending these constraints for embedding a VN with $K >= 2$ can result in combinatorial number of constraints, and can be investigated as a future research.

3.4 Heuristic Algorithm

The coordinated node and link mapping of the aforementioned ILP formulations without the disjointedness constraints is at least as hard as the $\mathcal{NP}$-Hard *Multi-commodity Unsplittable Flow Problem* [58], when the sources and destinations of the flows are unknown. To tackle the computational intractability of the ILP formulations, we develop a heuristic algorithm for the joint spare backup capacity allocation and survivable embedding problem against multiple (e.g., K) SLink failures. Our heuristic algorithm is presented as a pseudocode in Alg. 5. The algorithm solves the joint optimization problem for arbitrary K in two steps: (i) estimate the spare backup bandwidth on the VLinks, determine the disjointedness requirements based on this estimation, and perform a VN Embedding (§ 3.4.1), (ii) re-optimize spare backup bandwidth allocations using different backup multiplexing techniques proposed in [17, 76, 77] (§ 3.4.2).

3.4.1 Joint Spare Bandwidth Allocation and VN Embedding

Algorithm 4: VN Initialization

```
1  function VNInitialization(G, Ĝ, K)
2      D ← {d_1, d_2, ... d_|Ê|}
3      foreach û ∈ V̂ do nmap_û ← NIL
4      foreach (û, v̂) ∈ Ê do
5          S^est_ûv̂ ← 0, Λ_ûv̂ ← σ, SRG_ûv̂ ← d_1, emap_ûv̂ ← φ
6          T_ûv̂ ← Max − Priority − Queue()
7          foreach k=1, 2, ..., K do
8              backup^est_ûv̂(k) ← φ
9      V̂ ← Sort û ∈ V̂ in decreasing order of |N(û)|
10     return V̂
```

Alg. 5 first initializes the data structures for estimated spare capacity of the VLinks and the SRGs by invoking Alg. 4. To do so, Alg. 4 sets estimated spare backup bandwidth of each VLink, $S^{est}_{\hat{u}\hat{v}}$ to 0 and places all the VLinks into a single SRG d_1. In addition, it initializes a Max-Priority-Queue $T_{\hat{u}\hat{v}}$ for each $(\hat{u}, \hat{v})$ to be used for spare capacity computation. $T_{\hat{u}\hat{v}}$ will contain, in descending order, the bandwidths of all the VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$. Alg. 4 then sorts the VNodes in the VN from the most constrained to the least constrained ones, *i.e.*, in decreasing order of their degrees and returns the sorted list of VNodes as $\hat{\mathcal{V}}$. Note that if two VNodes have equal degrees then Alg. 4 arbitrarily selects one of them.

Algorithm 5: Embed VN with Protection

function *VNEmbedding(*G, $\hat{G}$, C, σ, K*)*
  $\hat{\mathcal{V}} \leftarrow$ VNInitialization(G, $\hat{G}$, K)
  foreach $\hat{u} \in \hat{\mathcal{V}}$ **do**
    foreach $\hat{v} \in \mathcal{N}(\hat{u})$ **do**
      foreach *k=1, 2, ..., K* **do**
        $backup^{est}_{\hat{u}\hat{v}}(k) \leftarrow$ GetBackup($\hat{G}, (\hat{u}, \hat{v}), \Lambda, emap, k$)
        foreach $(\hat{x}, \hat{y}) \in backup^{est}_{\hat{u}\hat{v}}(k)$ **do**
          $T_{\hat{x}\hat{y}}.EnQueue(b_{\hat{u}\hat{v}})$, $S^{est}_{\hat{x}\hat{y}} \leftarrow \sum_{0 \le i < \min(K,\ T_{\hat{x}\hat{y}}.Size())} T_{\hat{x}\hat{y}}[i]$
          if $SRG_{\hat{u}\hat{v}} = SRG_{\hat{x}\hat{y}}$ **then**
            Find $d_j \in D$ *s.t.* $SRG_{\hat{w}\hat{z}} \neq d_j, \forall(\hat{w}, \hat{z}) \in \hat{E}$
            $SRG_{\hat{u}\hat{v}} \leftarrow d_j$
          $\hat{\mathcal{H}}_{\hat{x}\hat{y}} \leftarrow \{(\hat{a}, \hat{b}) \in \hat{E} | (\hat{x}, \hat{y}) \in backup^{est}_{\hat{a}\hat{b}}(1)\}$
          foreach $(\hat{a}, \hat{b}) \in \hat{\mathcal{H}}_{\hat{x}\hat{y}}$ **do**
            if $SRG_{\hat{u}\hat{v}} = SRG_{\hat{a}\hat{b}}$ **then**
              Find $d_j \in D$ *s.t.* $SRG_{\hat{w}\hat{z}} \neq d_j, \forall(\hat{w}, \hat{z}) \in \hat{E}$
              $SRG_{\hat{u}\hat{v}} \leftarrow d_j$
          if $K > 1$ **then**
            foreach *j=1, 2, ..., k-1* **do**
              foreach $(\hat{a}, \hat{b}) \in backup^{est}_{\hat{u}\hat{v}}(j)$ **do**
                if $SRG_{\hat{x}\hat{y}} = SRG_{\hat{a}\hat{b}}$ **then**
                  Find $d_j \in D$ *s.t.* $SRG_{\hat{w}\hat{z}} \neq d_j, \forall(\hat{w}, \hat{z}) \in \hat{E}$
                  $SRG_{\hat{x}\hat{y}} \leftarrow d_j$
      $best_{\hat{u}} \leftarrow$ NIL, $Q^{best}_{\hat{u}} \leftarrow \phi, c_{best} \leftarrow \infty$
      foreach $l \in L(\hat{u})$ **do**
        foreach $\hat{v} \in \mathcal{N}(\hat{u})$ **do**
          $\zeta \leftarrow \{(u, v) \in E | SRG_{\hat{x}\hat{y}} \neq SRG_{\hat{u}\hat{v}} \wedge (u, v) \in emap_{\hat{x}\hat{y}}, \forall(\hat{x}, \hat{y}) \in \hat{E}\}$
          $W \leftarrow C$, **foreach** $(m, n) \in \zeta$ **do** $W_{mn} \leftarrow \infty$
          if $nmap_{\hat{v}} \neq NIL$ **then** $Q_{\hat{u}\hat{v}} \leftarrow$CWSP($G, l, nmap_{\hat{v}}, b_{\hat{u}\hat{v}}, W$)
          else $Q_{\hat{u}\hat{v}} \leftarrow \min_{\forall m \in L(\hat{v})} \{\text{CWSP}(G, l, m, b_{\hat{u}\hat{v}}, W)\}$
        if $\exists \hat{v} \in \mathcal{N}(\hat{u}) : Q_{\hat{u}\hat{v}} = \phi$ **then** $c \leftarrow \infty$
        else $c \leftarrow \sum_{\forall \hat{v} \in \mathcal{N}(\hat{u})} Q_{\hat{u}\hat{v}}$
        if $c < c_{best}$ **then** $best_{\hat{u}} \leftarrow l$, $Q^{best}_{\hat{u}} \leftarrow Q_{\hat{u}}, c_{best} \leftarrow c$
      if $best_{\hat{u}} = NIL$ **then return** $\{\phi, \phi, \phi, \phi\}$
      $nmap_{\hat{u}} \leftarrow best_{\hat{u}}$
      foreach $\hat{v} \in \mathcal{N}(\hat{u})$ ***and*** $nmap_{\hat{v}} \neq NIL$ **do**
        $emap_{\hat{u}\hat{v}} \leftarrow Q^{best}_{\hat{u}\hat{v}}$, $\Lambda_{\hat{u}\hat{v}} \leftarrow Cost(Q^{best}_{\hat{u}\hat{v}})$
  $\{backup, S\} \leftarrow$ UpdateBackup($\hat{G}$, $backup$, SRG, K)
  return $\{nmap, emap, backup, S\}$

Alg. 5 then proceeds to map the VNodes in the sorted order of $\hat{\mathcal{V}}$. For a VNode $\hat{u}$, Alg. 5 first finds K estimated backup VPaths for each VLink incident to $\hat{u}$ by iteratively invoking GetBackup procedure (Alg. 6). Alg. 6 invokes Constrained Weighted Shortest Path (CWSP) procedure to compute a VPath with at least $b_{\hat{u}\hat{v}}$ bandwidth between $\hat{u}$ and $\hat{v}$ in the VN $\hat{G}$, according to a weight function, $Weight^{est}$.

Alg. 6 first computes the weight function $Weight^{est}$ for all the VLinks and invokes CWSP procedure to obtain k_{th} backup VPath between $\hat{u}$ and $\hat{v}$. For the VLink $(\hat{u}, \hat{v})$, Alg. 6 assigns infinite weights to all the VLinks which have been used by the other backup VPaths of $(\hat{u}, \hat{v})$ to avoid having them appear again in the current backup VPath (Line 3). It gives lower weights to a VLink $(\hat{x}, \hat{y})$, if at least K VLinks in $\hat{\mathcal{H}}_{\hat{x}\hat{y}}$ have bandwidth larger than $b_{\hat{u}\hat{v}}$. This means that $S^{est}_{\hat{x}\hat{y}}$ is already sufficient to serve the bandwidth of K VLinks impacted by K SLink failures, thus allowing $(\hat{u}, \hat{v})$ to share the assigned spare bandwidth $S^{est}_{\hat{x}\hat{y}}$ without increasing it (Line 4). On the other hand, if $\hat{\mathcal{H}}_{\hat{x}\hat{y}}$ has less than K VLinks (i.e., $|T_{\hat{x}\hat{y}}| < K$) or if $b_{\hat{u}\hat{v}}$ is larger than at least one of the first K largest bandwidths of $\hat{\mathcal{H}}_{\hat{x}\hat{y}}$, $S^{est}_{\hat{x}\hat{y}}$ will increase as a result of using $(\hat{x}, \hat{y})$ in the kth backup VPath between $\hat{u}$ and $\hat{v}$. The increased spare capacity requirement is represented by $S^{temp}_{\hat{x}\hat{y}}$ and the amount of increase is $b_{\hat{u}\hat{v}}$ in the first case or the difference between $b_{\hat{u}\hat{v}}$ and the K_{th} element of $T_{\hat{x}\hat{y}}$ in the second case. The weight function of CWSP takes the possibility of increase in the spare capacity requirement and the mapping cost $\Lambda_{\hat{x}\hat{y}}$ of an already mapped VLink $(\hat{x}, \hat{y})$ into account and assigns $(\hat{x}, \hat{y})$ a weight proportional to both of these. In line 7, a special case occurs when a VLink $(\hat{x}, \hat{y})$ is not yet mapped. In this case, $\Lambda_{\hat{x}\hat{y}}$ is set to use the mean SPath length (σ) as an indicator of future cost (Line 5 of Alg. 4). Finally, an infinite weight is set to the VLinks whose mapped SPaths do not have adequate residual capacity to exclude them from the search space (Line 8 of Alg. 6).

After computing each estimated backup VPath $backup^{est}_{\hat{u}\hat{v}}(k)$, Alg. 5 updates $S^{est}_{\hat{x}\hat{y}}$ for all $(\hat{x}, \hat{y}) \in backup^{est}_{\hat{u}\hat{v}}(k)$. To do so, Alg. 5 first inserts $b_{\hat{u}\hat{v}}$ into the right position of the decreasingly sorted queue $T_{\hat{x}\hat{y}}$. Following (3.28), Alg. 5 computes $S^{est}_{\hat{x}\hat{y}}$ as the summation of either the first K largest elements of $T_{\hat{x}\hat{y}}$ or all the elements of $T_{\hat{x}\hat{y}}$ if $|T_{\hat{x}\hat{y}}| < K$ (Line 8). It then places $(\hat{u}, \hat{v})$ and $(\hat{x}, \hat{y})$ into different SRGs (Line 11). It also places $(\hat{u}, \hat{v})$ and all other VLinks that use $(\hat{x}, \hat{y})$ in their backup VPaths into different SRGs (Line 16). Finally, it places $(\hat{x}, \hat{y})$ and any VLink $(\hat{a}, \hat{b})$ present in any of the other $K - 1$ backup VPaths of $(\hat{u}, \hat{v})$ into different SRGs. After finding the estimated backup VPaths and SRGs of all the incident VLinks of a VNode $\hat{u}$, Alg. 5 finds the mapping of $\hat{u}$ and VLinks incident to $\hat{u}$. It iterates over all candidate SNodes $l \in L(\hat{u})$ and selects the one that results in the least cost mapping for all the VLinks incident to $\hat{u}$ (Line $24 - 34$). For a specific $l \in L(\hat{u})$ and $\hat{v} \in \mathcal{N}(\hat{u})$, if $\hat{v}$ is already mapped to $nmap_{\hat{v}}$, Alg. 5 computes CWSP from l to $nmap_{\hat{v}}$ (Line 28), while satisfying capacity constraints and SRG constraints in the SN (using the weights

Algorithm 6: Compute Backup VPath of a VLink

1 **function** *GetBackup*$(\hat{G}, (\hat{u}, \hat{v}),\ \Lambda,\ emap, k)$
2 **foreach** $(\hat{x}, \hat{y}) \in \hat{E}$ **do**
3 **if** $(\exists j | j < k\ and\ (\hat{x}, \hat{y}) \in backup^{est}_{\hat{u}\hat{v}}(j))$ **then** $Weight^{est}_{\hat{x}\hat{y}} \leftarrow \infty$
4 **else if** $T_{\hat{x}\hat{y}}[K-1] \geq b_{\hat{u}\hat{v}}$ **then** $Weight^{est}_{\hat{x}\hat{y}} \leftarrow 1$
5 **else if** $emap_{\hat{x}\hat{y}} = \phi$ ***or*** $\min_{(u,v)\in Q_{\hat{x}\hat{y}}} b^{residual}_{uv} \geq b_{\hat{u}\hat{v}}$ **then**
6 $S^{temp}_{\hat{x}\hat{y}} \leftarrow b_{\hat{u}\hat{v}} + \sum_{0\leq i<\min(K-1,\ T_{\hat{x}\hat{y}}.Size())} T_{\hat{x}\hat{y}}[i]$
7 $Weight^{est}_{\hat{x}\hat{y}} \leftarrow (S^{temp}_{\hat{x}\hat{y}} - S^{est}_{\hat{x}\hat{y}}) \times \Lambda_{\hat{x}\hat{y}}$
8 **else** $Weight^{est}_{\hat{x}\hat{y}} \leftarrow \infty$
9 $Weight^{est}_{\hat{u}\hat{v}} \leftarrow \infty$
10 **return** CWSP$(\hat{G}, \hat{u}, \hat{v}, b_{\hat{u}\hat{v}}, Weight^{est})$

in W). To do so, Alg. 5 identifies the set of SLinks that the mapping of $(\hat{u}, \hat{v})$ should be disjoint from and assigns ∞ as their weights (Line 27). On the other hand, if $\hat{v}$ is not mapped yet, it computes CWSPs from l to the SNodes $m \in L(\hat{v})$ and selects the CWSP with the minimum cost (Line 29). After mapping a VNode $\hat{u}$, Alg. 5 maps the VLinks whose both endpoints have already been mapped and updates Λ of the mapped VLinks (Line 36). Upon mapping all the VLinks of a VN, Alg. 5 returns *nmap*, *emap*, *backup*, and S representing the VNode mapping, VLink mapping, backup VPaths, and spare backup capacities, respectively.

3.4.2 Reconfiguring the Allocated Spare Backup Bandwidth

The last phase (Alg. 7) of our heuristic employs different techniques to further optimize the spare capacity allocation. Since Alg. 5 performs spare capacity assignment and embedding of VLinks sequentially based on some estimation, it is possible that spare capacity of initial VLinks may have been allocated using partial backup VPath selection and VLink embedding information. Once complete information is available, sharing of spare capacity can be further enhanced by taking into account SRGs of different failure scenarios and backup VPath multiplexing combinations [76, 77]. Due to space constraints, Alg. 7 illustrates the spare capacity optimization for only single (i.e., $K = 1$) and double (i.e., $K = 2$) link failure scenarios. Techniques similar to [76, 77] can be adopted to optimize spare capacity for higher (i.e., $K > 2$) failure scenarios albeit the expected huge number of SRG and VPath multiplexing combinations.

Algorithm 7: Reconfigure Spare Capacities of all VLinks

function *UpdateBackup*$(\hat{G}, backup, SRG, K)$
  if $K = 1$ **then**
    foreach $(\hat{u}, \hat{v}) \in \hat{E}$ **do** $S_{\hat{u}\hat{v}} \leftarrow 0$
    $\hat{R} \leftarrow$ longest cycle in $\hat{G}$ such that no VLink pair in $\hat{R}$ shares an SLink on their mapped SPaths
    foreach $(\hat{x}, \hat{y}) \in \hat{R}$ **do**
      $S_{\hat{x}\hat{y}} \leftarrow \max_{\forall(\hat{u},\hat{v})\in\hat{R}}\{b_{\hat{u}\hat{v}}\}$
    foreach $(\hat{u}, \hat{v}) \in \hat{E}$ **do**
      foreach $(\hat{x}, \hat{y}) \in \hat{E} \setminus \{(\hat{u}, \hat{v})\}$ **do**
        if $SRG_{\hat{u}\hat{v}} = SRG_{\hat{x}\hat{y}}$ **then** $Weight_{\hat{x}\hat{y}} \leftarrow \infty$
        else if $S_{\hat{x}\hat{y}} \geq b_{\hat{u}\hat{v}}$ **then** $Weight_{\hat{x}\hat{y}} \leftarrow 1$
        else $Weight_{\hat{x}\hat{y}} \leftarrow (b_{\hat{u}\hat{v}} - S_{\hat{x}\hat{y}})$
      $Weight_{\hat{u}\hat{v}} \leftarrow \infty$
      $backup_{\hat{u}\hat{v}} \leftarrow \text{CWSP}(\hat{G}, \hat{u}, \hat{v}, b_{\hat{u}\hat{v}}, Weight)$
      foreach $(\hat{x}, \hat{y}) \in backup_{\hat{u}\hat{v}}$ **do**
        $S_{\hat{x}\hat{y}} \leftarrow \max(S_{\hat{x}\hat{y}}, b_{\hat{u}\hat{v}})$
  else if $K = 2$ **then**
    foreach $((\hat{u}, \hat{v}), (\hat{a}, \hat{b})) \in \hat{E} \times \hat{E}$ *s.t.* $(\hat{u}, \hat{v}) \neq (\hat{a}, \hat{b})$ **do**
      if $(\{(\hat{u}, \hat{v}) \cap backup^{est}_{\hat{a}\hat{b}}(1)\} = \phi$ *and* $\{(\hat{a}, \hat{b}) \cap backup^{est}_{\hat{u}\hat{v}}(1)\} = \phi$ *and* $\{Q_{\hat{u}\hat{v}} \cap Q_{\hat{a}\hat{b}}\} = \phi)$ **then**
        foreach $(\hat{x}, \hat{y}) \in backup^{est}_{\hat{u}\hat{v}}(1)$ **do**
          if $(\hat{x}, \hat{y}) \in backup^{est}_{\hat{a}\hat{b}}(2)$ **then**
            $T_{\hat{x}\hat{y}}.DeQueue(\min(b_{\hat{u}\hat{v}}, b_{\hat{a}\hat{b}}))$
            $S_{\hat{x}\hat{y}} \leftarrow \sum_{0\leq i<\min(K,\ T_{\hat{x}\hat{y}}.Size())} T_{\hat{x}\hat{y}}[i]$
        foreach $(\hat{x}, \hat{y}) \in backup^{est}_{\hat{u}\hat{v}}(2)$ **do**
          if $(\hat{x}, \hat{y}) \in backup^{est}_{\hat{a}\hat{b}}(1)$ **then**
            $T_{\hat{x}\hat{y}}.DeQueue(\min(b_{\hat{u}\hat{v}}, b_{\hat{a}\hat{b}}))$
            $S_{\hat{x}\hat{y}} \leftarrow \sum_{0\leq i<\min(K,\ T_{\hat{x}\hat{y}}.Size())} T_{\hat{x}\hat{y}}[i]$
        foreach $(\hat{x}, \hat{y}) \in backup^{est}_{\hat{u}\hat{v}}(2)$ **do**
          if $(\hat{x}, \hat{y}) \in backup^{est}_{\hat{a}\hat{b}}(2)$ **then**
            $T_{\hat{x}\hat{y}}.DeQueue(\min(b_{\hat{u}\hat{v}}, b_{\hat{a}\hat{b}}))$
            $S_{\hat{x}\hat{y}} \leftarrow \sum_{0\leq i<\min(K,\ T_{\hat{x}\hat{y}}.Size())} T_{\hat{x}\hat{y}}[i]$
  return $\{backup, S\}$

For single SLink failure, Alg. 7 leverages p-cycle based protection to optimize the spare backup bandwidth $S_{\hat{u}\hat{v}}$ for each mapped VLink $(\hat{u}, \hat{v}) \in \hat{E}$ [17]. Alg. 7 first finds the longest

cycle $\hat{R}$ in $\hat{G}$ such that no pairs of VLinks in $\hat{R}$ share an SLink in their mappings (Line 4). Recall from § 3.2.4 that each $(\hat{x}, \hat{y}) \in \hat{R}$ belongs to distinct SRGs for $K = 1$. Therefore, Alg. 7 allocates the maximum of the demands of all the VLinks in $\hat{R}$ to each $S_{\hat{x}\hat{y}} \in \hat{R}$ (Line 6). It then recomputes backup VPath $backup_{\hat{u}\hat{v}}$ for each $(\hat{u}, \hat{v}) \in \hat{E}$ using a process similar to Alg. 6. However, Alg. 7 utilizes VLink embedding information to better compute the backup VPaths. It does so by setting ∞ as the weight of the VLink $(\hat{x}, \hat{y})$ if $(\hat{u}, \hat{v})$ and $(\hat{x}, \hat{y})$ are in the same SRG (Line 9). Alg. 7 also enhances the spare capacity sharing by setting unit weights to the VLinks having already assigned spare capacities (Line 10). Alg. 7 then invokes CWSP procedure with the weight function to compute $backup_{\hat{u}\hat{v}}$ (Line 13). Finally, $S_{\hat{x}\hat{y}}$ for each $(\hat{x}, \hat{y}) \in backup_{\hat{u}\hat{v}}$ is updated accordingly (Line 15).

For two SLink failures, Alg. 7 utilizes backup VPath multiplexing to optimize spare backup bandwidth allocation [76, 77]. Backup multiplexing is based on the observation that not all backup VPaths are used simultaneously for certain failure scenarios, thus allowing to share the spare bandwidth allocated to the common VLinks in those VPaths. For instance, if two VLinks $(\hat{u}, \hat{v}) \in \hat{E}$ and $(\hat{a}, \hat{b}) \in \hat{E}$ do not share any SLink in their SPaths and none of $(\hat{u}, \hat{v})$ and $(\hat{a}, \hat{b})$ is present in the first backup VPaths of the other one, two SLink failures can only impact either both $(\hat{u}, \hat{v})$ and $(\hat{a}, \hat{b})$ or one of $(\hat{u}, \hat{v})$ and $(\hat{a}, \hat{b})$ and one of the VLinks in the first backup VPaths of $(\hat{u}, \hat{v})$ and $(\hat{a}, \hat{b})$. In the first case, both the first VPaths of $(\hat{u}, \hat{v})$ and $(\hat{a}, \hat{b})$ are used together, whereas only the second VPath of either $(\hat{u}, \hat{v})$ or $(\hat{a}, \hat{b})$ is used in the second case. Therefore, the first VPath of $(\hat{u}, \hat{v})$ (or $(\hat{a}, \hat{b})$) and the second VPath of $(\hat{a}, \hat{b})$ (or $(\hat{u}, \hat{v})$) are not used together. The same is true for both the second VPaths of $(\hat{u}, \hat{v})$ and $(\hat{a}, \hat{b})$. Alg. 7 exploits such observations by reducing the spare bandwidths of the common VLinks in those VPaths that will not be used simultaneously for two SLink failure scenarios (Line $17 - 30$).

3.4.3 Running Time Analysis

CWSP procedure is the core of both Alg. 5 and Alg. 7. CWSP procedure is implemented using a modified *Dijkstra's shortest path* algorithm that takes into account the constraints and weights as depicted above. *Dijkstra's* algorithm using a min-priority queue on G runs in $O(|E| + |V| \log |V|)$ time. *CWSP* is invoked $O(|\hat{V}|\mathcal{L}\delta^2)$ times in Alg. 5, where $\mathcal{L}$ and δ are the maximum size of a location constraint set and maximum degree of a VNode, respectively. Therefore, the overall running time of the heuristic is $O(|\hat{V}|\mathcal{L}\delta^2(|E| + |V| \log |V|))$.

3.5 Evaluation

We first describe the simulation setup and compared approaches in § 3.5.1. The performance metrics are defined in § 3.5.2. Finally, we describe our evaluation results focusing on: (i) evaluation for single link failure scenarios (§ 3.5.3), (ii) performances of double link failure scenarios (§ 3.5.5), and (iii) Comparison with Existing SN level survivability (§ 3.5.4).

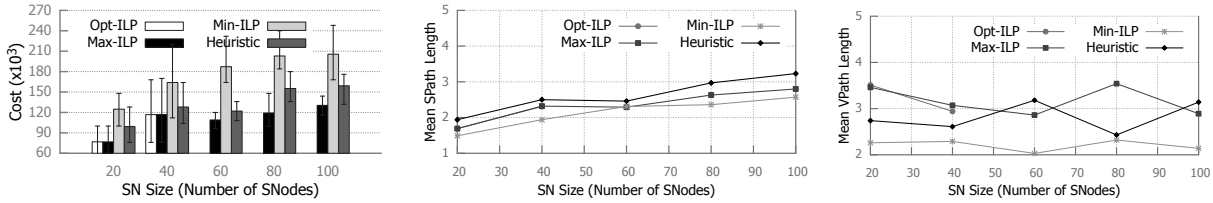
3.5.1 Simulation Setup

We implement *Opt-ILP*, *Max-ILP*, and *Min-ILP* from § 3.3 using IBM ILOG CPLEX libraries and compare them with a C++ implementation of the heuristic algorithm, referred as *Heuristic* from now on. For each simulation run, we generate an SN and 5 random VNs. For each SN, we take the mean value of each performance metric over the 5 VNs. For single failure scenarios ($K = 1$), SN and VN size is varied between 20 and 100 nodes and 3 and 11 nodes, with increments of 10 and 2, respectively. We change the connectivity of both SNs and VNs by varying the link to node ratio (LNR) from 1.12 to 3.00 and from 1.0 to 2.17, respectively to match with realistic topologies [67, 153]. We make sure VNs are 2- and 3-edge connected to survive single and double link failures, respectively. For double failure cases ($K = 2$), we start with VNs of size 4 since there is no simple graph of size less than 4 that has 3-edge connectivity. For $K = 2$, we vary VN sizes on an SN with size 25 and LNR 1.8. We also compare performances of *Heuristic* for $K = 1$ and $K = 2$ in large scale topologies. In these cases, VN sizes vary between 10 and 100 nodes with 21 and 285 links on SNs with 500 and 1000 nodes with 2016 and 4022 links. Given the number of nodes and links of a problem instance, the source and destination of an SLink or a VLink are decided randomly. Location constraints of VNodes are chosen randomly, and VLink demands are set to 10% of the SLink bandwidths. Simulations are performed on a machine with 2×8-core 2.0 Ghz Intel Xeon E5-2650 processors and 256GB of RAM.

3.5.2 Performance Metrics

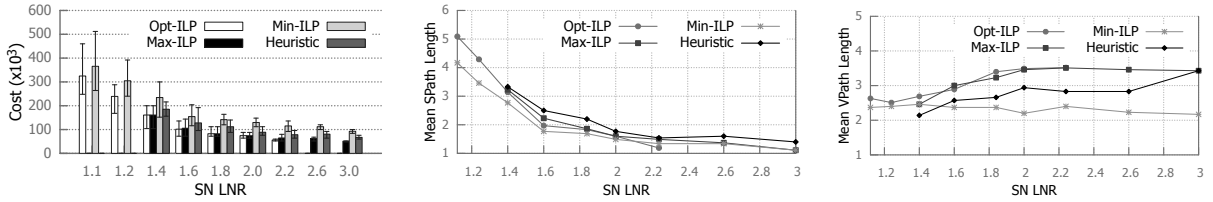
Cost

The cost of embedding a VN computed using (3.20). We set a unit cost for allocating bandwidth on an SLink, therefore, (3.20) directly represents resource consumption.



(a) Cost Analysis (b) SPath Length Analysis (c) VPath Length Analysis

Figure 3.4: Impact of SN Size on Single Link Failure Protection



(a) Cost Analysis (b) SPath Length Analysis (c) VPath Length Analysis

Figure 3.5: Impact of SN Density on Single Link Failure Protection

Mean SPath Length

The mean length of the SPath used to map a VLink of a VN.

Mean VPath Length

The mean length of the VPath used as a backup path for a VLink in a VN.

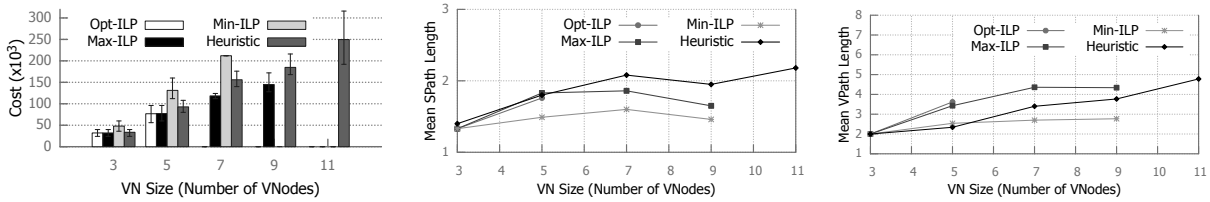
Execution Time

The time required for an algorithm to find an embedding of a VN.

3.5.3 Evaluation Results for Single SLink Failure Scenarios

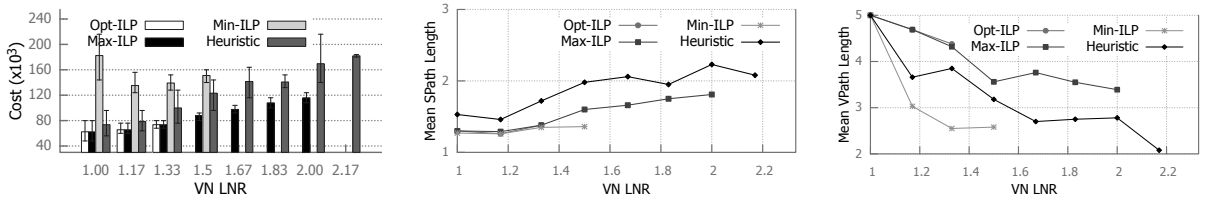
Impact of SN Topology

Fig. 3.4 presents performance metrics for different SN sizes, while keeping the VN size and SN LNR fixed at 5 and 1.8, respectively. Since SNode degrees remain the same, SN



(a) Cost Analysis (b) SPath Length Analysis (c) VPath Length Analysis

Figure 3.6: Impact of VN Size on Single Link Failure Protection

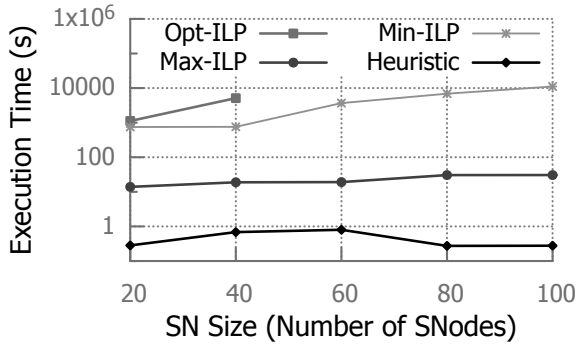


(a) Cost Analysis (b) SPath Length Analysis (c) VPath Length Analysis

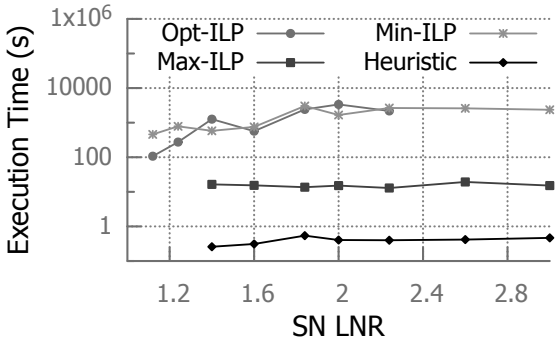
Figure 3.7: Impact of VN Density on Single Link Failure Protection

diameters increase with increasing SN size. We see that embedding costs in Fig. 3.4(a) increase for all the compared approaches with increasing SN size. This stems from the fact that the location constraints of the VNodes of a VN are chosen to be far apart from one another in an SN with larger diameter. This behavior is verified in Fig. 3.4(b) that shows the increase in SPath lengths with the increase in SN sizes. Longer SPaths require higher amount of substrate resources for embedding the VLinks of a VN, hence, the higher cost. Another takeaway from Fig. 3.4(b) is that mean SPath lengths of *Min-ILP* and *heuristic* are the lowest and highest, respectively. *Min-ILP* selects shorter SPaths for embedding VLinks because of the lower number of disjoint path constraints imposed by the shorter VPaths as shown in Fig. 3.4(c). *Min-ILP* prefers the shorter VPaths since it has to allocate dedicated spare capacity on all the VLinks in a VPath. In contrast, all other approaches choose longer VPaths to maximize the spare capacity sharing on VLinks.

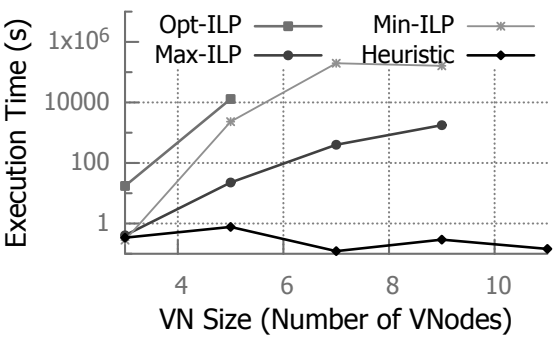
Fig. 3.5 presents the impact of SN density on the performance metrics by varying SN LNRs. For very sparse SNs, *i.e.*, SNs with very low LNRs, *Max-ILP* and *heuristic* fail to find sufficient disjoint SPaths, imposed by the SRG constraints, resulting in infeasible solutions. However, *Opt-ILP* and *Min-ILP* are able to find solutions with very high costs by reducing the number of SRGs (Fig. 3.5(a)). Lower number of SRGs, in turn, reduces the opportunities of spare capacity sharing on VPaths. Despite reducing SRGs, *Opt-ILP* and *Min-ILP* still have to find disjoint SPaths to satisfy survivability constraints. In a sparse SN, a set of disjoint SPaths between pairs of SNodes becomes significantly longer than



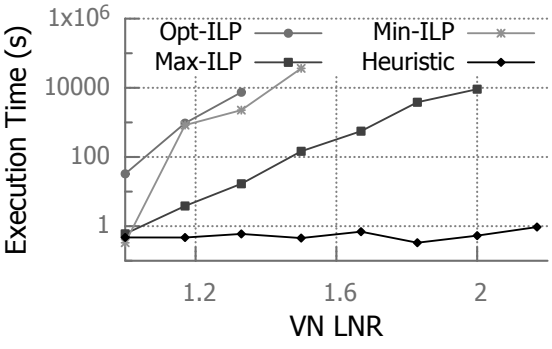
(a) Varying SN Size



(b) Varying SN Density

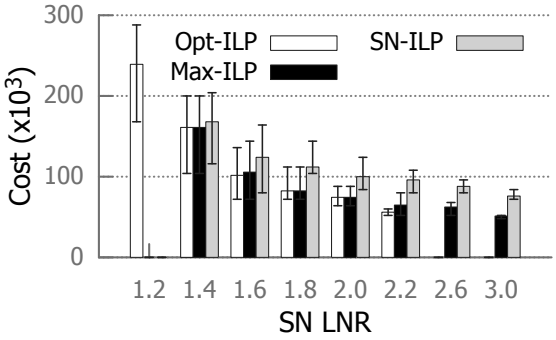


(c) Varying VN Size

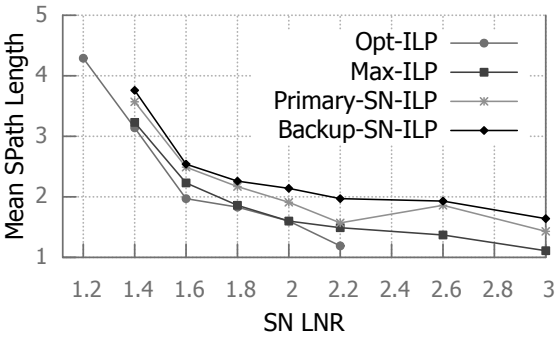


(d) Varying VN Density

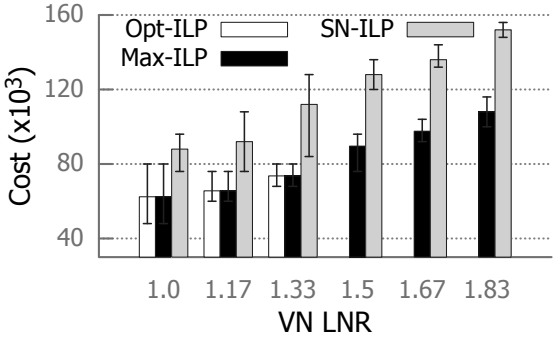
Figure 3.8: Scalability Analysis for Single Link Failure Protection



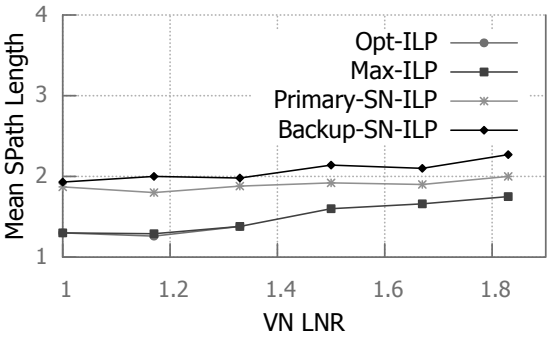
(a) Cost Analysis



(b) SPath Length Analysis

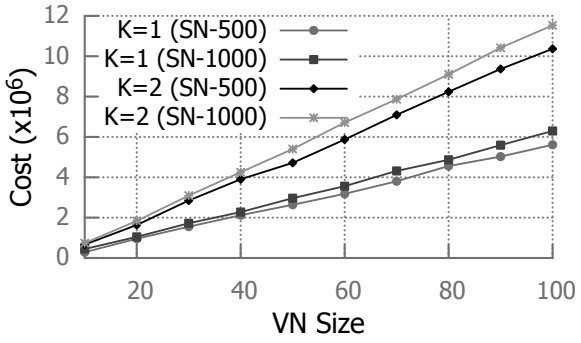


(c) Cost Analysis

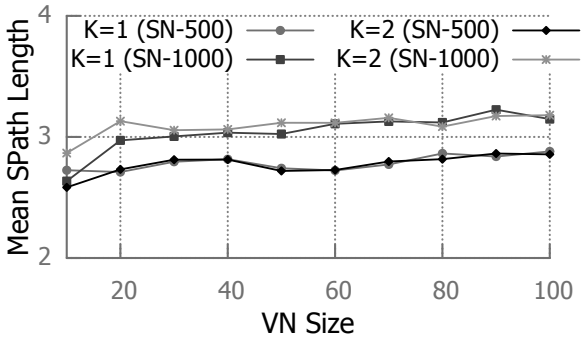


(d) SPath Length Analysis

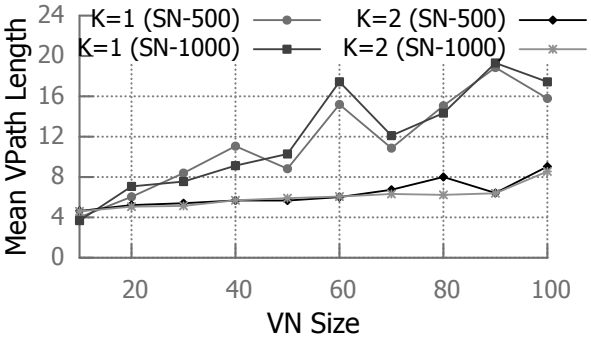
Figure 3.9: Comparison with SN level Survivability of [43]



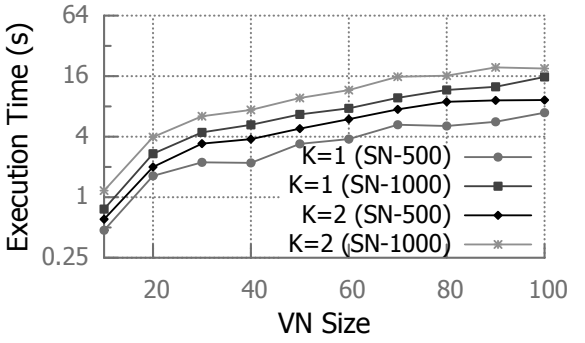
(a) Cost Analysis



(b) SPath Length Analysis



(c) VPath Length Analysis



(d) Execution Time Analysis

Figure 3.10: Performance of Heuristic with Single ($K = 1$) and Double ($K = 2$) Link Failure Protections on Large Scale Test Cases

the set of non-disjoint SPaths between the same pairs of SNodes. These aforementioned factors contribute to the higher embedding costs for sparse SNs. As SN LNR increases, the number and length of the SPaths (also disjoint SPaths) increase and decrease, respectively, hence the decrease in mean SPath lengths in Fig. 3.5(b). As a consequence, cost decreases for all the approaches as shown in Fig. 3.5(a). Fig. 3.5(c) shows that mean VPath lengths for all the approaches except *Min-ILP* increase initially with increasing SN LNR. When VPath lengths get closer to the VN diameter in these approaches, they remain almost constant with increasing SN LNR. The initial increase in mean VPath length is due to the use of the same VLinks by more VPaths, leading to more spare capacity sharing. However, sparse SNs cannot satisfy the SRG constraints imposed by longer VPaths, therefore, all the approaches except *Min-ILP* select shorter VPaths. Since *Min-ILP* does not allow any spare capacity sharing, there is little impact of SN density on VPath lengths.

Impact of VN Topology

Fig. 3.6 presents performance metrics for different VN sizes, while keeping the SN size, SN LNR, and VN LNR fixed at 50, 1.82, and 1.4, respectively. Fig. 3.6(a) shows that cost increases as VN size gets larger. This is partially due to the allocation of more SN resources for embedding higher number of VLinks of a larger VN. The other contributor is the longer embedding SPath as shown in Fig. 3.6(b). SPath length increases with VN size to find the higher number of disjoint SPaths that are required to satisfy the higher number of SRG and survivability constraints induced by longer VPaths shown in Fig. 3.6(c). In case of *Min-ILP*, VPath lengths increase due to rise in VN diameter resulting from increase in VN size. The other approaches select even longer VPaths to enable more spare capacity sharing on the VLinks of the selected VPaths.

Fig. 3.7 compares performance metrics by varying the VN LNR, while keeping the SN size, SN LNR, and VN size fixed at 50, 1.82, and 6, respectively. The takeaway from Fig. 3.7(a), Fig. 3.7(b), and Fig. 3.7(b) is that both cost and mean SPath length increase with increasing VN LNR, whereas mean VPath length shows an opposite trend. We explain these behaviors as follows. The sparsest connected VN is a ring topology. In such a topology, each VLink $(\hat{u}, \hat{v})$ has exactly one VPath that contains all the other VLinks in the ring except $(\hat{u}, \hat{v})$, and all the VLinks in the ring have to be embedded disjointedly to satisfy the survivability constraints. Hence, all the approaches have the same mean VPath length for the sparsest VN, *i.e.*, VN with the lowest LNR. The results of ring VNs also illustrate that *Min-ILP* allocates ~ 3 times extra resources to guarantee the same level of survivability provided by the other approaches. As VN LNR increases, *Min-ILP* prefers the shortest possible VPaths to minimize the amount of additional resources. Other approaches

strike a balance between reducing disjoint path requirements for satisfying survivability constraints and increasing spare capacity sharing as well as disjoint path requirements of SRG constraints induced by shorter and longer VPaths, respectively. Hence, the mean VPath lengths of all the approaches leveraging spare capacity sharing decrease much slower than those of *Min-ILP*, which does not allow any sharing. Despite a decrease in VPath lengths as VN LNR increases, all the approaches except *Min-ILP* have to satisfy higher number of SRG constraints to enable more spare capacity sharing. This contributes to the increasing trend of cost and mean SPath length.

Discussion of results for *Max-ILP* and *Min-ILP*

As we see in Fig. 3.4-Fig. 3.7, *Max-ILP* closely approximates *Opt-ILP*, whereas *Min-ILP* provides the upper bound of cost. The proximity between *Max-ILP* and *Opt-ILP* can be explained using the analysis presented in § 3.3.3 and mean SPath length values. Since SPath lengths of *Max-ILP* are always less than twice the SPath lengths of *Min-ILP*, *Max-ILP* yields the lower cost. Hence, for embedding a VLink, *Opt-ILP* prefers spare capacity sharing of *Max-ILP* than dedicated spare capacity allocation of *Min-ILP*. Despite the sequential nature of *Heuristic*, its performance remains closer to those of *Max-ILP* than *Min-ILP*. Specifically, compared to *Max-ILP*, *Heuristic* provisions ∼21% additional resources on average.

Scalability Analysis

Fig. 3.8 demonstrates the scalability of the compared approaches by plotting execution times in logarithmic scale. As we can see, *Opt-ILP* only scales to very small networks. The reduction in problem complexities in terms of constraints and variables in *Max-ILP* and *Min-ILP* are reflected in their ability to scale to larger problem instances than *Opt-ILP*. Among them *Min-ILP* has to satisfy the lowest number of disjoint path constraints. It explores a much larger solution space than *Max-ILP*, requiring more time. Following our discussion on problem complexities in § 3.3.3, VN dimension has a more profound impact on the scalability of the ILP-based approaches than SN dimension. We observe this impact in Fig. 3.8(c) and Fig. 3.8(d) that show, with our current hardware, the ILP-based approaches hit ceilings in terms of VN size or VN LNR. Even for the successful cases, the ILP-based approaches require several orders of magnitude more time to solve similar problem instances compared to *Heuristic* that can scale to much larger problem instances.

3.5.4 Comparison with Existing SN Level Survivability

One of the key motivations for embedding a VN augmented with spare capacity is the hypothesis that more resource efficient embedding is possible with protection at the VN layer compared to doing the same at the SN layer. We provide a quantitative comparison to support this hypothesis. For this scenario, we have explored the SVNE literature to find a representative VN embedding approach that adopts SN level shared spare capacity allocation and found the proposal in [43] as the closest match. However, [43] only provides a heuristic algorithm that yields suboptimal solutions. Hence, we have formulated an ILP, namely *SN-ILP*, following the concepts presented in [43] for solving the exact problem of VN embedding with SN layer shared spare capacity allocation and used it as a baseline for comparison. We implement *SN-ILP* using IBM ILOG CPLEX library. For each VLink in a VN, *SN-ILP* allocates SN resources on a primary and a backup SPath disjoint from the primary. However, the spare capacity allocated on the backup SPaths can be shared among the VLinks whose primary SPaths are edge disjoint. For a fair comparison, we juxtapose the performance metrics of *SN-ILP* with those of our ILP formulations that share spare capacity, *i.e.*, *Opt-ILP* and *Max-ILP* in Fig. 3.9. Fig. 3.9(a) reveals that, similar to *Max-ILP*, *SN-ILP* yields infeasible solutions for sparse SNs due to lack of path diversity. In all the cases when *SN-ILP* is able to find a solution as shown in Fig. 3.9(a) and Fig. 3.9(c), *SN-ILP* incurs higher costs than both *Opt-ILP* and *Max-ILP*. In these cases, *SN-ILP* allocates, on average, ∼33% additional resources compared to *Max-ILP*. We explain this behavior with the help of Fig. 3.9(b) and Fig. 3.9(d). These figures show that both primary and backup SPath lengths of *SN-ILP* exceed SPath lengths of *Opt-ILP* and *Max-ILP*. This is due to the fact that finding a pair of disjoint SPaths between a pair of SNodes is more difficult than finding an SPath that is disjoint from a set of SPaths between different set of SNodes [84]. Another reason for lower costs of *Opt-ILP* and *Max-ILP* compared to *SN-ILP* is that *Opt-ILP* and *Max-ILP* employ more spare capacity sharing on the VPaths than spare capacity sharing on the SPaths employed by *SN-ILP*.

3.5.5 Evaluation Results for Multiple Link Failure Scenarios

Small Scale Performance

Table 3.1 compares the performances of ILP formulation presented in § 3.3.4 and *Heuristic* for $K = 2$. As evident from the table, the ILP formulation does not scale beyond 5 node VNs due to the combinatorial number of constraints for $K = 2$. On the other hand, *Heuristic* scales to larger topologies at the expense of incurring higher cost. As seen in

Table 3.1: Evaluation results for double ($K = 2$) link failures

VNode Count	Cost (ILP)	Cost (Heuristic)	SPath Length (ILP)	SPath Length (Heuristic)	VPath Length (ILP)	VPath Length (Heuristic)
4	112×10^3	127×10^3	1.57	1.77	2.1	2.0
5	158×10^3	184×10^3	1.65	1.93	2.47	2.25
6	-	247×10^3	-	2.0	-	2.4

the table, the higher cost of *Heuristic* than ILP is attributed to the longer SPaths for embedding VLinks and less sharing of spare capacity along shorter VPaths. Nonetheless, the average cost of *Heuristic* remains within 20% of the ILP.

Large Scale Performance

In Fig. 3.10, we present the performances of the *Heuristic* with $K = 1$ and $K = 2$ for large scale topologies, where multiple failures are more likely to occur [67, 65]. Fig. 3.10(a) shows that cost increases proportionally with the increase in VN size, following the same trend observed in Fig. 3.6(a). The $\sim$ 100% increase in embedding cost of $K = 2$ from $K = 1$ is due to the increase in spare backup bandwidth requirements to survive double link failures. The slightly higher embedding cost for an SN with 1000 nodes than for an SN with 500 nodes in both $K = 1$ and $K = 2$ cases is due to the increase in SN diameter similar to the cases in Fig. 3.4(a). The increase in SN diameter is reflected in Fig. 3.10(b) that shows SPaths are longer in a 1000 node SN than in the 500 node one. Different behavior is observed in Fig. 3.10(c) that shows VPath lengths of $K = 1$ are much larger than those of $K = 2$. This is due to adopting different reconfiguration strategies for different failure scenarios by Alg. 7. For $K = 1$, the goal of Alg. 7 is to assign spare bandwidth along the longest p-cycle, thus resulting in longer VPaths. Such strategy can be expensive for $K > 1$, since K disjoint p-cycles are needed to survive K SLink failures. Furthermore, p-cycle based strategy offers less room for spare capacity sharing for $K > 1$, as two VLinks on the p-cycle are on the backup VPaths of each other. Therefore, for $K = 2$, Alg. 7 prefers shorter VPaths as shown in Fig. 3.10(c) and optimizes spare capacity through backup multiplexing technique described in Alg. 7. Finally, Fig. 3.10(d) shows that our *Heuristic* is able to find solutions within a reasonable time limit in compliance with the running time analysis presented in § 3.4.3.

3.6 Conclusion

In this chapter, we have studied the SVNE problem that arises in network virtualization *a.k.a.* network slicing from a new perspective. Instead of managing failure survivability at the SN level, we have proposed to delegate the failure management tasks to the VN level in support of more autonomous VNs. Hence, a new variety of SVNE problem emerges that requires a VN to be augmented with sufficient spare backup capacity and embedded on the SN accordingly to ensure survivability against multiple substrate link failures. For the case of single substrate link failure, we have formulated the optimal solution to this joint optimization problem as a QIP and transformed it into an ILP. We have presented simplified ILP formulations to solve special cases of the problem and presented a mathematical study to analyze the impact of SN topology on the level of spare capacity sharing. We have then extended the most resource efficient ILP formulation for single link failure to handle the case of multiple substrate link failures. We have also proposed a heuristic algorithm to tackle the computational complexity of the ILP formulations. The heuristic algorithm provides a generic framework to survive multiple link failures and demonstrates spare capacity optimization techniques for single and double link failure scenarios.

We have performed simulations to evaluate our solutions for single and double link failure scenarios. Simulation results show that ILP formulations for the two special cases can provide upper and approximately lower bounds of the solution spectrum. Moreover, the heuristic allocates $\sim$21% additional resources compared to the approximated lower bound, while executing several orders of magnitude faster. Large scale simulations of the heuristic algorithm shows that protection against double link failures comes at the cost of nearly two times of the resources incurred by protection against single link failures. A quantitative comparison between the SVNE with VN level protection and the traditional SVNE with SN level protection reveals that the former can save $\sim$33% resources on average, compared to those required by the latter.

Chapter 4

Recovery from Node Failure in Virtual Network Embedding

With increasing deployments of virtual networks (VNs) in commercial-grade networks with commodity hardware, VNs need to tackle failures in the underlying substrate network. In this chapter, we study the problem of recovering a batch of VNs affected by a substrate node failure. A number of mechanisms have been proposed to increase VN reliability against substrate resource failures. These mechanisms can be broadly classified into two categories [78]: a) proactively provision disjoint redundant resources as backup [71, 131] and b) reactively re-embed the failed nodes and links of a VN on the available resources after a failure has occurred [37, 25]. Proactive approaches offer immediate recovery from failures at the expense of backup resource reservation [70, 82, 88]. However, preallocating backup resources for multiple failures resulting from a substrate node failure can be extremely expensive [82, 177]. Instead, an SP may prefer to reactively re-embed the failed part of its VN to avoid the huge cost of preallocated backup resources in a failure-prone SN. The SP can adopt a load balanced VN embedding strategy to leave higher amount of available resources during re-embedding. Moreover, in case of permanent substrate node failures (*e.g.*, hardware malfunction), the nodes and links of all the affected VNs have to be re-embedded. These scenarios motivate us to study the problem of re-embedding a batch of VNs affected by a substrate node failure.

While VN embedding is already intractable, the combinatorial number of sequences of VNs in a batch re-embedding further increases the complexity [152]. In addition, any solution must be significantly fast (*e.g.*, in the order of milliseconds) to meet the stringent timing requirement usually imposed by Service Level Agreements (SLAs). To meet such requirements, we consider the re-embedding of only the failed nodes and links of a batch

of VNs without disrupting their unaffected parts. While re-embedding the failed part as well as the unaffected part of an active VN may incur lower costs of re-embedding [37, 32], it will require additional time for virtual node migration and virtual link re-configuration, which may increase VN downtime. Further, the reactive approaches in [25], [80] opt for the recovery of all the failed nodes and links of a VN. If the SN does not have adequate resources to recover all the failed nodes and links, those approaches re-embed the complete VN, turning it inactive for a while and causing service disruptions.

Unlike existing approaches, we adopt a generalized recovery approach that allows partial recovery of an affected VN, while adhering to any SLA requirement. To demonstrate the versatility of our approach, we investigate two different recovery models. The first one is a fair recovery model (FRM) that maximizes the number of recoveries across all the affected VNs. This can be the sought-after choice of an InP who wants to treat all the affected VNs fairly in a resource constrained SN. Then, we explore a priority-based recovery model (PRM) that takes into account an InP's preference during recovery. PRM allows an InP to prioritize the recovery of affected VNs based on SLA strictness, impacts of failure, profits, and so on to achieve its goal. It is worth discussing here that a partial recovery scheme may lead to skewness in the utilization of SN resources. To eliminate such skewness, VN reconfiguration mechanisms such as [49, 118, 161, 160] should be performed periodically during an off-peak period.

In this chapter, we focus on the problem of **Re**covering from a **No**de failure in **V**irtu**a**l Ne**t**work **E**mbedding (*ReNoVatE*). It accepts a batch of VN failures resulting from a single substrate node failure, and produces alternate embeddings for the failed virtual nodes and links. The objective of FRM is to maximize the number of recovered virtual links across all the affected VNs, while minimizing total bandwidth required for recovery. On the other hand, PRM seeks to prioritize the recovery of affected VNs based on some predefined requirements and minimize total bandwidth for recovery. We formulate *ReNoVatE* as an Integer Linear Programming (ILP) based optimization model, namely *Opt-ReNoVatE*. Since *Opt-ReNoVatE* cannot scale to large instances of the problem, we devise an efficient heuristic algorithm, called *Fast-ReNoVatE*, to find satisfactory solutions within prescribed time limits. We evaluate *Fast-ReNoVatE* by extensive simulations and compare it with *Opt-ReNoVatE*, as well as with the most related state-of-the-art proposal in the literature [25]. Evaluation results demonstrate that *Fast-ReNoVatE* performs close to *Opt-ReNoVatE* and outperforms the state-of-the-art solution in terms of i) number of recovered virtual links, ii) cost of recovery, and iii) execution time.

The rest of the chapter is organized as follows. Section 4.1 presents the related literature. In Section 4.2, we present the system model and formally introduce the problem. Section 4.3 presents two variants of *Opt-ReNoVatE* for solving the problem optimally.

The heuristic solution, *Fast-ReNoVatE*, is presented in Section 4.4. Evaluation results are presented in Section 4.5. Finally, we summarize our findings and conclude in Section 4.6.

4.1 Related Work

Survivability of VNs during substrate failures was first addressed by Rahman *et al.*, in [132]. They formulated the problem of ensuring survivability in VNs under single SLink failure as a Mixed Integer Linear Program and proposed heuristics to obtain solutions in a reasonable time. A large body of research literature has been developed since to address different aspects of SVNE such as considering single SLink failure [71, 94], multiple SLink failures [141], single SNode failure [177, 43], regional failures [176, 112, 128], and ensuring certain levels of availability [129, 179, 89] among others. The approaches for ensuring survivability of VNs during substrate failures can be broadly classified into two categories, namely, proactive and reactive approaches. Proactive approaches provision redundant backup resources when a VN is embedded in the first place, whereas, reactive approaches take mitigation actions after a failure has occured. In the following we discuss the proactive (§ 4.1.1) and reactive (§ 4.1.2) approaches from SVNE literature and contrast them with our solution for *ReNoVatE*.

4.1.1 Pro-active Approaches

Pro-active approaches for SVNE preallocates backup resources to ensure Quality of Service for the VNs during one or more substrate failures. Most of the approaches focusing on substrate node (SNode) failure are pro-active [70, 82, 177]. Yu *et al.*, , [177] proposed a two-step method to recover a VN. The first step enhances the VN with backup virtual nodes (VNodes) and virtual links (VLinks), and the second step maps this enhanced VN on the SN. This approach, in the worst case, has to reserve a backup VNode for each VNode. In contrast, [70] designed the enhanced VN with a failure-dependent strategy to reduce backup resources. Despite the resource efficiency of this approach, it is not practical due to the large number of migrations of working VNodes. Unlike these methods, [82] presented a joint optimization strategy for allocating primary and backup resources altogether. The location constrained SVNE, to address geographically-correlated SNode failures, has been studied in [176, 88, 81]. While [81] adopts sequential embedding of working and backup VNodes, [176, 88] embeds them jointly to minimize total bandwidth. Recently, [48, 47] proposed embedding primary and dedicated backup resources for each VNode and VLink in a VN simultaneously. There is a growing trend towards designing survivable resource

allocation schemes for embedding virtual data centers (VDCs) in cloud [11], [22], [174]. For instance, [171] proposed a scheme for provisioning VDCs with backup virtual machines and links. Bodik *et al.*, , [26] proposed an optimization framework for improving survivability, while reducing total bandwidth consumption. Yeow *et al.*, , [174] defines the reliability level as a function of backup resources that are shared between VNs through opportunistic pooling. Finally, [179] provided a VDC embedding framework for achieving high VDC availability by considering heterogeneous failure rates.

Preallocated backup resources remain unused during normal operations and hence reduce resource utilization. In contrast, ReNoVatE takes a reactive approach and allocates resources only when a failure has occurred, thus improving on the resource utilization.

4.1.2 Reactive Approaches

Reactive approaches, on the other hand, do not preallocate any backup resources. Chang *et al.*, , [37] proposed a migration aware VN re-embedding algorithm to recover from an SNode failure. The algorithm allows the migration of some active VNodes and VLinks to free up some substrate resources, thus facilitating the re-mapping of the failed VNodes and VLinks. Cai *et al.*, [32] addressed the problem of optimally upgrading the existing VN in a highly evolving SN. Their goal is to minimize the upgrading cost, in addition to migration and remapping costs, while satisfying QoS constraints. Both of these approaches may need a chain of migrations to converge, thus disrupting ongoing communication in the VN. Reactive approaches to recover from geographically correlated failures has been proposed in [112] and [128]. However, unlike ReNoVatE, [112, 128] allow the VNs to operate at a degraded Quality of Service. Recently, reactive recovery approaches have been proposed for multi-cast VNs. Unlike traditional VNs, multi-cast VNs take the Quality of Service requirements of multi-cast services (*e.g.*, bounded delay for multi-cast completion) into account. Ayoubi *et al.*, has shown that restoring multi-cast trees with delay constraints during a single substrate node failure is NP-hard for general graphs [19]. Polynomial time heuristic algorithms for restoring multi-cast trees in VNs during single substrate node, single substrate link and multiple substrate node and link failures have been proposed in [18], [63], and [64], respectively. In contrast to the reactive approaches for restoring multi-cast VNs, our proposal does not make any assumption about the structure of SN and communication pattern in the VN. Another line of works [132, 123], originated in optical networks, proposed a hybrid mechanism to select from a set of precomputed detours for recovery during failure. However, in a highly saturated SN, this mechanism may not find adequate resources left for the recovery.

Distributed reactive approaches including, [80, 150], proposed multi-agent based algorithms to dynamically adapt the VNs in response to SN failures. When an agent detects a failure of another agent in the same cluster, the agents within the same cluster collaborate with each other to re-provision the failed VNodes and VLinks. These approaches may generate sub-optimal solutions due to the lack of global knowledge of the SN. Finally, [25] proposed a greedy algorithm to find alternate substrate resources for the affected part of a VN. In case of resource inadequacy, this approach requires remapping the entire VN resulting in prolonged service unavailability. Nonetheless, existing reactive algorithms focus on re-embedding only a single VN, whereas *ReNoVatE*, for the first time, considers partial re-embedding of a set of affected VNs to improve recovery performance.

4.2 System Model and Problem Statement

We first present a mathematical representation of a substrate network, virtual network, and types of failure (§ 4.2.1). Then, we formally define the problem (§ 4.2.2). A glossary of key notations used in the chapter is provided in Table 4.1.

4.2.1 System Model

Substrate Network (SN)

We represent the SN as an undirected graph, $G = (V, E)$, where V and E denote the set of SNodes and Substrate Links (SLinks), respectively. The set of neighbors of an SNode $u \in V$ is denoted by $\mathcal{N}(u)$. Bandwidth capacity and residual bandwidth of an SLink $(u, v) \in E$ are represented by b_{uv} and r_{uv}, respectively, while the cost of allocating one unit of bandwidth in (u, v) is C_{uv}. V^f and E^f represent the set of failed SNodes and SLinks, respectively. P_{uv} represents a path between SNodes u and v

Virtual Network (VN)

We denote the set of VNs embedded on the SN G as $\bar{G} = \{\bar{G}_1, \bar{G}_2, \ldots \bar{G}_{|\bar{G}|}\}$. Each VN $\bar{G}_i \in \bar{G}$ is represented as an undirected graph $\bar{G}_i = (\bar{V}_i, \bar{E}_i)$, where $\bar{V}_i$ and $\bar{E}_i$ are the sets of VNodes and VLinks of $\bar{G}_i$, respectively. The set of neighbors of a VNode $\bar{u} \in \bar{V}_i$ is denoted by $\mathcal{N}(\bar{u})$. Each VLink $(\bar{u}, \bar{v}) \in \bar{E}_i$ has a bandwidth demand $b_{i\bar{u}\bar{v}}$. We associate a penalty $\pi_{i\bar{u}\bar{v}}$ to each VLink $(\bar{u}, \bar{v}) \in \bar{E}_i$, where $\pi_{i\bar{u}\bar{v}}$ represents the penalty due to resource unavailability

of $(\bar{u}, \bar{v})$. Each VN $\bar{G}_i$ has a set of location constraints, $L_i = \{L_i(\bar{u}) | L_i(\bar{u}) \subseteq V, \forall \bar{u} \in \bar{V}_i\}$, such that a VNode $\bar{u} \in \bar{V}_i$ can only be mapped to an SNode $u \in L_i(\bar{u})$. We represent this location constraint with a binary variable $\ell_{i\bar{u}u}$, defined as:

$$\ell_{i\bar{u}u} = \begin{cases} 1 & \textit{iff } \bar{u} \in \bar{V}_i \text{ can be provisioned on } u \in V, \\ 0 & \text{otherwise.} \end{cases}$$

Types of failure

Let, $f(\bar{u})$ and $g(\bar{u}\bar{v})$ denote the SNode and substrate path where $\bar{u}$ and $(\bar{u}, \bar{v})$ have been embedded, respectively. An SNode failure results in a set of VNode and VLink failures of a VN $\bar{G}_i$ defined as $\bar{V}_i^f = \{\bar{u} \in \bar{V}_i | f(\bar{u}) \subseteq V^f\}$ and $\bar{E}_i^f = \{(\bar{u}, \bar{v}) \in \bar{E}_i | (u, v) \in g(\bar{u}\bar{v}) \wedge (u, v) \in E^f\}$, respectively. There are two types of VLinks in $\bar{E}_i^f$:

Adjacent VLinks: The set of VLinks adjacent to the failed VNode $\bar{u} \in \bar{V}_i^f$ is represented by $\bar{\mathcal{E}}_i^f = \{(\bar{u}, \bar{v}) | \bar{u} \in \bar{V}_i^f \wedge \bar{v} \in \mathcal{N}(\bar{u})\}$.

Independent VLinks: The set of VLinks that have failed due to the failure of some SLinks on their mapped substrate paths is denoted by $\bar{\mathtt{E}}_i^f = \{(\bar{u}, \bar{v}) | (u, v) \in g(\bar{u}\bar{v}) \wedge (u, v) \in E^f \wedge \bar{u} \notin \bar{V}_i^f \wedge \bar{v} \notin \bar{V}_i^f\}$.

Finally, $\bar{V}^f = \{\bigcup \bar{V}_i^f\}$, $\bar{E}^f = \{\bigcup \bar{E}_i^f\}$, and $\bar{\mathtt{E}}^f = \{\bigcup \bar{\mathtt{E}}_i^f\}$ represent the set of failed VNodes, VLinks, and Independent VLinks of all the VNs in $\bar{G}$, respectively. Fig. 4.1 illustrates the embedding of two VNs, $\bar{G}_1$ with $\bar{V}_1 = \{a, b, c\}$ and $\bar{G}_2$ with $\bar{V}_2 = \{d, e\}$ on the SN G shown in the bottom. The numbers next to a VLink and an SLink represent the VLink demand and SLink residual bandwidth, respectively. The VNode mapping *i.e.*, $f(.)$ is shown by placing a VNode beside its mapped SNode and the VLink mapping *i.e.*, $g(.)$ is depicted by dashed paths between SNodes. For instance, $f(a) = \{D\}$, $f(b) = \{C\}$, $f(c) = \{G\}$, $f(d) = \{B\}$, $f(e) = \{H\}$ and $g(ab) = \{DB, BC\}$, $g(ac) = \{DH, HG\}$, $g(bc) = \{CF, FG\}$, and $g(de) = \{BD, DH\}$. We now show the impact of an SNode failure with $V^f = \{D\}$ and $E^f = \{DB, DE, DH\}$. VN $\bar{G}_1$ experiences a VNode failure with $\bar{V}_1^f = \{a\}$. Consequently, the VLinks adjacent to a fail leading to $\bar{\mathcal{E}}_1^f = \{ab, ac\}$. Note there is no Independent VLink failures in $\bar{G}_1$, and so $\bar{E}_1^f = \bar{\mathcal{E}}_1^f$. On the other hand, $\bar{V}_2^f = \phi$ since no VNode of $\bar{G}_2$ is mapped on D. However, the failure of D results in an independent VLink failure yielding $\bar{E}_2^f = \bar{\mathtt{E}}_2^f = \{de\}$. Hence, any recovery algorithm should re-embed all the affected VNodes and VLinks from $\bar{G}_1$ and $\bar{G}_2$ leaving unaffected part such as VLink bc undisrupted.

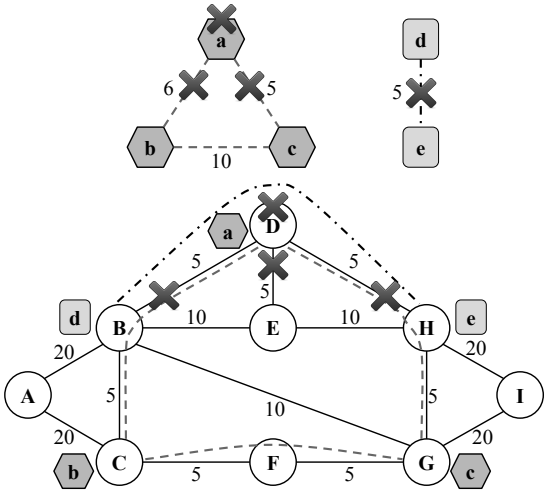


Figure 4.1: VN embedding and impact of failure

4.2.2 Problem Statement

Given an SN $G = (V, E)$, a failed SNode implying $|V_f| = 1$, and a set of affected VNs $\bar{G}$ embedded on G, re-embed the failed VNodes in $\bar{V}^f$ and the failed VLinks in $\bar{E}^f$ on G such that the re-embedding achieves the following objectives:

- Primary objective differs based on the recovery model being chosen. For FRM, primary objective is to maximize the total number of recovered VLinks across all the affected VNs. In PRM, primary objective is to minimize the total penalty for all the failed VLinks that remain unrecovered.
- Secondary objective is to minimize the total cost of re-embedding in terms of SLink bandwidth consumption.

subject to the following constraints:

- a failed VNode $\bar{u} \in \bar{V}_i^f$ is re-embedded on exactly one SNode, $v \in L_i(\bar{u})$. In addition, multiple VNodes of the same VN cannot be mapped to an SNode. However, multiple VNodes from different VNs can share an SNode.
- a failed VLink $(\bar{u}, \bar{v}) \in \bar{E}_i^f$ is re-embedded on a substrate path $P_{f(\bar{u})f(\bar{v})}$ having sufficient bandwidth to accommodate the demand of the VLink. The re-embedding cannot use a substrate path containing the failed SNode.

- VNodes and VLinks not affected by the SNode failure are not re-embedded.

4.3 ILP Formulation: *Opt-ReNoVatE*

We provide an ILP formulation, *Opt-ReNoVatE*, based on the *Multi-commodity Flow Problem* formulation of *ReNoVatE*. We first present the decision variables (§ 4.3.1). Then, we introduce the constraints (§ 4.3.2) followed by the objective functions of the two variants of *Opt-ReNoVatE* (§ 4.3.3 and § 4.3.4). Finally, we describe the complexity of the problem in § 4.3.5.

4.3.1 Decision Variables

The following decision variables indicate VNode and VLink embedding of a VN $\bar{G}_i \in \bar{G}$ on an SN G.

$$y_{i\bar{u}u} = \begin{cases} 1 & \textit{iff } \bar{u} \in \bar{V}_i \text{ is mapped to } u \in V, \\ 0 & \text{otherwise.} \end{cases}$$

$$x_{uv}^{i\bar{u}\bar{v}} = \begin{cases} 1 & \textit{iff } (\bar{u}, \bar{v}) \in \bar{E}_i \text{ is mapped to } (u, v) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

The objective of *Opt-ReNoVatE* is to recover as many failed VLinks in $\bar{E}^f$ as possible to mitigate the impact of failure. It may be possible that not all VLinks in $\bar{E}^f$ can be re-embedded due to substrate resource limitation. The following decision variable defines which VLinks are re-embedded:

$$z_{i\bar{u}\bar{v}} = \begin{cases} 1 & \textit{iff } (\bar{u}, \bar{v}) \in \bar{E}_i^f \text{ is mapped to any substrate path} \\ 0 & \text{otherwise.} \end{cases}$$

Table 4.1: Summary of Key Notations

$G = (V, E)$	Substrate Network
b_{uv}	Bandwidth capacity of SLink $(u, v) \in E$
r_{uv}	Residual bandwidth of SLink $(u, v) \in E$
V^f	Set of failed SNodes
E^f	Set of failed SLinks
C_{uv}	Cost of unit bandwidth on SLink $(u, v) \in E$
$\bar{G}_i = (\bar{V}_i, \bar{E}_i)$	i-th Virtual Network Request
$b_{i\bar{u}\bar{v}}$	Bandwidth requirement of virtual link $(\bar{u}, \bar{v}) \in \bar{E}_i$
$L_i(\bar{u})$	Location constraint set for virtual node $\bar{u} \in \bar{V}_i$
$\ell_{i\bar{u}u} \in \{0, 1\}$	$\ell_{i\bar{u}u} = 1$ if $u \in L_i(\bar{u})$, $u \in V, \bar{u} \in \bar{V}_i$
$f(\bar{u}) \in V$	Embedding of VNode $\bar{u}$
$g(\bar{u}\bar{v}) \in 2^E$	Embedding of VLink $(\bar{u}, \bar{v})$
$\bar{V}_i^f$	Set of failed VNodes from VN $\bar{V}_i$
$\bar{E}_i^f$	Set of failed VLinks from VN $\bar{V}_i$
$\bar{\mathcal{E}}_i^f$	Set of failed VLinks adjacent to a failed VNode
$\bar{\mathtt{E}}_i^f$	Set of VLinks not adjacent to a failed VNode
$x_{uv}^{i\bar{u}\bar{v}} \in \{0, 1\}$	$x_{uv}^{i\bar{u}\bar{v}} = 1$ if $(u, v) \in E$ is on the embedded substrate path for $(\bar{u}, \bar{v}) \in \bar{E}_i$
$y_{i\bar{u}u} \in \{0, 1\}$	$y_{i\bar{u}u} = 1$ if $\bar{u} \in \bar{V}_i$ is mapped to $u \in V$
$z_{i\bar{u}\bar{v}} \in \{0, 1\}$	$z_{i\bar{u}\bar{v}} = 1$ *iff* $(\bar{u}, \bar{v}) \in \bar{E}_i^f$ is mapped to any substrate path
$\pi_{i\bar{u}\bar{v}}$	Penalty due to resource unavailability of $(\bar{u}, \bar{v}) \in \bar{E}_i^f$

4.3.2 Constraints

Intactness of Unaffected VNodes and VLinks

The mapping of VNodes and VLinks that are not affected by the substrate failure remains unchanged. Constraints (4.1) and (4.2) ensure that unaffected VNodes and VLinks are not re-embedded.

$$\forall \bar{G}_i \in \bar{G}, \forall \bar{u} \in \bar{V}_i \setminus \bar{V}_i^f : y_{i\bar{u}f(\bar{u})} = 1 \tag{4.1}$$

$$\forall \bar{G}_i \in \bar{G}, \forall (\bar{u}, \bar{v}) \in \bar{E}_i \setminus \bar{E}_i^f, \forall (u, v) \in g(\bar{u}\bar{v}) : x_{uv}^{i\bar{u}\bar{v}} = 1 \tag{4.2}$$

Exclusion of Failed SNodes and SLinks from Re-embedding

The failed VNodes or VLinks cannot use any of the failed SNodes or SLinks during re-embedding. Constraint (4.3) ensures that the failed VNodes are not re-embedded on the failed SNodes, and (4.4) ensures that the failed VLinks are not re-embedded on substrate paths containing a failed SLink.

$$\forall \bar{G}_i \in \bar{G}, \forall \bar{u} \in \bar{V}_i^f, \forall u \in V_f : y_{i\bar{u}u} = 0 \tag{4.3}$$

$$\forall \bar{G}_i \in \bar{G}, \forall (\bar{u}, \bar{v}) \in \bar{E}_i^f, \forall (u, v) \in E_f : x_{uv}^{i\bar{u}\bar{v}} = 0 \tag{4.4}$$

Link Mapping Constraints

Constraint (4.5) prevents overcommitment of SLink bandwidth. Constraint (4.6) ensures that the in-flow and out-flow of each SNode is equal except at the SNodes where the endpoints of a failed VLink are embedded. Finally, constraint (4.7) ensures that if a VLink $(\bar{u}, \bar{v})$ is selected to be re-embedded due to the failure of $\bar{u}$, there is some flow from the SNode u where $\bar{v}$ is embedded already.

$$\forall(u,v) \in E : \sum_{\forall \bar{G}_i \in \bar{G}} \sum_{\forall(\bar{u},\bar{v}) \in \bar{E}_i} x_{uv}^{i\bar{u}\bar{v}} \times b_{i\bar{u}\bar{v}} \leq b_{uv} \tag{4.5}$$

$$\forall \bar{G}_i \in \bar{G}, \forall(\bar{u},\bar{v}) \in \bar{E}_i^f, \forall u \in V \setminus f(\bar{v}) : \sum_{\forall v \in \mathcal{N}(u)} (x_{uv}^{i\bar{u}\bar{v}} - x_{vu}^{i\bar{u}\bar{v}}) \leq y_{i\bar{u}u} - y_{i\bar{v}u} \tag{4.6}$$

$$\forall \bar{G}_i \in \bar{G}, \forall(\bar{u},\bar{v}) \in \bar{E}_i^f, \forall u \in f(\bar{v}) : \sum_{\forall v \in \mathcal{N}(u)} (x_{uv}^{i\bar{u}\bar{v}} - x_{vu}^{i\bar{u}\bar{v}}) = z_{i\bar{u}\bar{v}} \tag{4.7}$$

Node Mapping Constraints

First, constraint (4.8) ensures that re-embedding of a failed VNode should be done according to the provided location constraint set. Second, constraint (4.9) makes sure that a VNode should be mapped to at most an SNode in the SN. Third, constraint (4.10) enforces that an SNode will not host more than one VNodes from the same VN. Finally, constraint (4.11) ensures that if a VLink $(\bar{u},\bar{v}) \in \bar{\mathcal{E}}_i^f$ is selected to be re-embedded due to the failure of $\bar{u}$, the VNode $\bar{u}$ must be re-embedded on an SNode according to the location constraint. Here, λ is a very large integer that turns the left side of (4.11) into a fraction between 0 and 1 when any of the $z_{i\bar{u}\bar{v}}$ is 1. This enforces the right side of (4.11) to become 1, thus ensuring the failed VNode to be re-embedded.

$$\forall \bar{G}_i \in \bar{G}, \forall \bar{u} \in \bar{V}_i^f, \forall u \in V : y_{i\bar{u}u} \leq \ell_{i\bar{u}u} \tag{4.8}$$

$$\forall \bar{G}_i \in \bar{G}, \forall \bar{u} \in \bar{V}_i^f, : \sum_{u \in V} y_{i\bar{u}u} \leq 1 \tag{4.9}$$

$$\forall \bar{G}_i \in \bar{G}, \forall u \in V : \sum_{\bar{u} \in \bar{V}_i} y_{i\bar{u}u} \leq 1 \tag{4.10}$$

$$\forall \bar{G}_i \in \bar{G}, \forall \bar{u} \in \bar{V}_i^f : \frac{1}{\lambda} \sum_{\bar{v} \in \mathcal{N}(\bar{u})} z_{i\bar{u}\bar{v}} \leq \sum_{\forall u \in V} y_{i\bar{u}u} \tag{4.11}$$

4.3.3 Fair Recovery Model

The fair recovery model (FRM) treats all the failed VLinks equally while recovering them. Hence, the objective of this model is to recover as many failed VLinks as possible. Following the problem statement, the objective function (4.12) of FRM has two components. The first and primary component maximizes the number of re-embedded failed VLinks. In other words, it minimizes the total number of un-recovered VLinks as shown in (4.12). It is obvious to observe that the minimum value of the primary component is zero. However, there can be multiple solutions that achieve the minimum value. Hence, we need a secondary component in the objective function to break ties among multiple solutions having the same value for the primary objective. Therefore, the secondary component minimizes the total cost of provisioning bandwidth for re-embedding the failed VLinks on substrate paths. A weight factor w is multiplied to the second component to impose the relative weight to the components of (4.12). The value of w is chosen to be a very small fraction so that it comes into effect only to break ties among multiple solutions that have the same value for the primary objective. In this way, w prefers the number of recovered VLinks over the cost of re-embedding.

$$\begin{aligned} \text{minimize} \left(|\bar{E}^f| - \sum_{\forall \bar{G}_i \in \bar{G}} \sum_{\forall (\bar{u},\bar{v}) \in \bar{E}_i^f} z_{i\bar{u}\bar{v}} \right) \\ + w \left(\sum_{\forall \bar{G}_i \in \bar{G}} \sum_{\forall (\bar{u},\bar{v}) \in \bar{E}_i^f} \sum_{\forall (u,v) \in E} x_{uv}^{i\bar{u}\bar{v}} \times C_{uv} \times b_{i\bar{u}\bar{v}} \right) \end{aligned} \tag{4.12}$$

4.3.4 Priority-based Recovery Model

The priority-based recovery model (PRM) prioritizes some failed VLinks over others to satisfy the SLA requirements or to minimize the impact of failure or to maximize profits. To impose such priorities, we utilize the penalty parameter $\pi_{i\bar{u}\bar{v}}$ associated to each VLink $(\bar{u}, \bar{v}) \in \bar{E}_i^f$. Each $\pi_{i\bar{u}\bar{v}}$ can be given as an input based on SLA requirement violation penalty or can be computed based on the impact of failure. For instance, a VLink with strict SLA requirement may have a higher value of the penalty than that of a VLink with less-stringent SLA requirement. The objective of PRM is to minimize the total penalty across all the failed VLinks. Similar to the FRM case, the objective function (4.13) has two components. The primary component minimizes the total penalty incurred by the VLinks that are not recovered. The second one minimizes the total cost of provisioning bandwidth

for re-embedding the failed VLinks on substrate paths. A weight factor w is multiplied to the second component to impose the relative weight to the components of (4.13). The value of w is chosen to be a very small fraction so that it comes into effect only to break ties among multiple solutions that have the same value for the primary objective. In this way, w prioritizes minimizing total penalty over the cost of re-embedding.

$$\begin{aligned} \text{minimize} & \left(\sum_{\forall \bar{G}_i \in \bar{G}} \sum_{\forall (\bar{u},\bar{v}) \in \bar{E}_i^f} (1 - z_{i\bar{u}\bar{v}}) \times \pi_{i\bar{u}\bar{v}} \right) \\ + w & \left(\sum_{\forall \bar{G}_i \in \bar{G}} \sum_{\forall (\bar{u},\bar{v}) \in \bar{E}_i^f} \sum_{\forall (u,v) \in E} x_{uv}^{i\bar{u}\bar{v}} \times C_{uv} \times b_{i\bar{u}\bar{v}} \right) \end{aligned} \tag{4.13}$$

Note that, if we assume $\pi_{i\bar{u}\bar{v}} = 1, \forall \bar{G}_i \in \bar{G}, \forall (\bar{u}, \bar{v}) \in \bar{E}_i$, (4.13) reduces to (4.12). Therefore, objective function of PRM *i.e.*, (4.13) encompasses objective functions of both FRM and PRM, and can be considered as the generalized recovery model.

4.3.5 Complexity of the Problem

The binary nature of the decision variables and the flow constraints of *Opt-ReNoVatE* prevent any VLink from being mapped to multiple substrate paths. This restricts the re-embedding of Independent VLinks to the $\mathcal{NP}$-hard *Multi-commodity Unsplittable Flow Problem* [58]. The VNode re-embedding follows from VLink re-embedding since there are no costs associated with VNodes. Therefore, the re-embedding of a VNode and its adjacent VLinks of a VN becomes the $\mathcal{NP}$-hard *Single-source Unsplittable Flow Problem* with unknown source [54]. When there are a batch of affected VNs, computing the best sequence of VNs from a combinatorial number of sequences to maximize the number of recovered VLinks makes the problem computationally intractable.

4.4 Heuristic Solution: *Fast-ReNoVatE*

Due to the intractability of *Opt-ReNoVatE*, we resort to a heuristic algorithm, *Fast-ReNoVatE*, to find feasible solutions in reasonable time. *Fast-ReNoVatE* re-embeds the failed VNodes and their Adjacent VLinks (§ 4.4.1) and Independent VLinks (§ 4.4.2) of the affected VNs efficiently.

4.4.1 Recovery of VNodes and Adjacent VLinks

The inputs to the VNode Recovery algorithm (Alg. 8) comprise an SN G, a set of affected VNs $\bar{G}$ that are embedded on G, the recovery model to be applied, and a set of failed SNodes V^f and failed SLinks E^f. The purpose of the recovery model is to select the proper model between the two possibilities *i.e.*, FRM and PRM. Alg. 8 initially sorts the affected VNs in $\bar{G}$ and $\bar{\mathcal{G}}$ represents this sorted order. The sorting function depends on the recovery model being adopted. In FRM, the function sorts the affected VNs in $\bar{G}$ in increasing order of the total lost bandwidth in their Adjacent VLinks. The total lost bandwidth of a VN, $\bar{G}_i \in \bar{\mathcal{G}}$ is computed as the summation of the bandwidth demands for all the failed VLinks in $\bar{\mathcal{E}}_i^f$ of $\bar{G}_i$. On the other hand, PRM sorts the affected VNs in $\bar{G}$ in decreasing order of the total penalty in their Adjacent VLinks. The total penalty of a VN, $\bar{G}_i \in \bar{\mathcal{G}}$ is computed as the summation of the penalty $\pi_{i\bar{u}\bar{v}}$ of all the failed VLinks in $\bar{\mathcal{E}}_i^f$ of $\bar{G}_i$. Intuitively, the algorithm proceeds to recover the VNs in $\bar{G}$ in the sorted order of $\bar{\mathcal{G}}$ to increase the number of recovered VLinks in FRM or to decrease the cumulative penalty in PRM. For each VN $\bar{G}_i$, the algorithm tries to re-embed the failed VNode $\bar{u} \in \bar{V}_i^f$ and the VLinks adjacent to $\bar{u}$ *i.e.*, $\bar{\mathcal{E}}_i^f$ to an SNode present in the location constraint set of $\bar{u}$ *i.e.*, $L_i(\bar{u})$ and to substrate paths, respectively. To accomplish this, it iterates over all the SNodes $l \in L_i(\bar{u}) \setminus V^f$, and selects the SNode, $best_{\bar{u}}$ that maximizes the cardinality of the set of substrate paths, $\mathcal{P}_l$, computed for the VLinks in $\bar{\mathcal{E}}_i^f$. In case of a tie, the SNode with the lower cost of the paths in $\mathcal{P}_l$, denoted by M, is selected. Finally, the algorithm re-embeds $\bar{u}$ and the VLinks in $\bar{\mathcal{E}}_i^f$ to $best_{\bar{u}}$ and to the paths in M, respectively.

As discussed in § 4.3.5, optimally computing the set of substrate paths $\mathcal{P}_l$ from an SNode $l \in L_i(\bar{u}) \setminus V^f$ for the VLinks in $\bar{\mathcal{E}}_i^f$ of a VN is $\mathcal{NP}$-hard. Majority of the VN embedding proposals aims to minimize the cost of embedding [59]. They would embed each VLink in $\bar{\mathcal{E}}_i^f$ one-by-one by adopting a minimum cost substrate path finding approach. However, in a bandwidth constrained scenario, a minimum cost path may contain some bottleneck SLinks. Allocating the bandwidth of these SLinks to a VLink may leave later VLinks unrecoverable. The objective of *ReNoVatE* is to maximize the number of recovered VLinks irrespective of the cost of recovery. Hence, our heuristic (Alg. 9) simultaneously computes $\mathcal{P}_l$ for all the VLinks in $\bar{\mathcal{E}}_i^f$ to maximize the cardinality of $\mathcal{P}_l$. Alg. 9 works by finding maximum flow from a source to a sink in a graph avoiding any bottleneck SLink. If we always send unit flow from a source to a sink in a graph, the paths carrying the maximum amount of flow will correspond to the maximum number of paths between the source and the sink without exceeding the capacity of the SLinks.

To implement this idea, Alg. 9 first augments the SN graph G with a pseudo sink SNode, namely S. It then adds a pseudo SLink from each SNode that hosts a VNode,

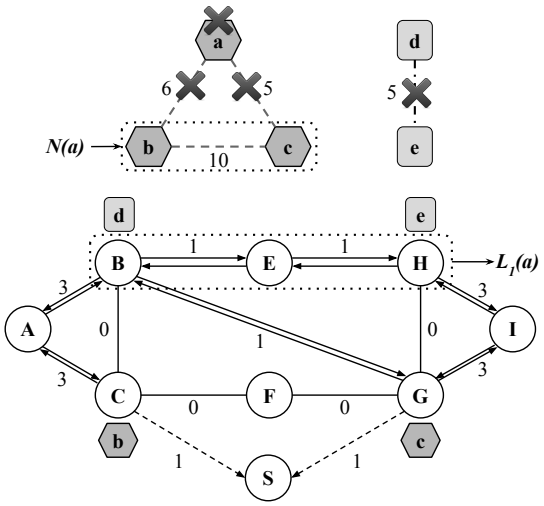


Figure 4.2: Maxflow Realization

$\bar{v} \in \mathcal{N}(\bar{u})$ to S, where $\bar{u} \in \bar{V}_i^f$. The capacity of each augmented SLink is set to 1 to ensure the un-splittability of the substrate paths. Further, each bidirectional SLink in $E \setminus E^f$ is replaced with two unidirectional SLinks, and the capacity of these new SLinks are discretized according to an estimation function, $\frac{r_{uv}}{\max\limits_{\forall\{(\bar{u},\bar{v})\in\bar{\mathcal{E}}_i^f\}}\{b_{i\bar{u}\bar{v}}\}}$. This stringent estimation ensures that each selected SLink can provide the bandwidth even for the VLink with the maximum demand among all the VLinks in $\bar{\mathcal{E}}_i^f$. Other estimation functions such as *min* or *average* could be used allowing over-subscription of bandwidth for some of the VLinks. Fig. 4.2 illustrates the maximum flow realization for the recovery of VN $\bar{G}_1$ embedded on SN G according to Fig. 4.1. Upon the failure of SNode D, VNode a and VLinks ab and ac of $\bar{G}_1$ fail. Since $\mathcal{N}(a) = \{b, c\}$, $f(b) = C$, and $f(c) = G$, we add SLinks CS and GS to sink S with capacity 1, as shown by the dashed arrows in Fig. 4.2. This figure depicts the transformed SN after removing the failed SNode and SLinks, replacing each bidirectional SLink with two unidirectional SLinks, and estimating the capacity of these SLinks using $\frac{r_{uv}}{\max\{6,5\}}$.

After augmenting and initializing the capacity and flow of each SLink in G, Alg. 9 proceeds with the steps of the *Edmonds-Karp* algorithm [56] for computing the maximum flow from each $l \in L_i(\bar{u}) \setminus V^f$ to S. We modify the *Edmonds-Karp* algorithm to compute the set of augmenting paths $\mathcal{P}$ from each l to S so that the sum of flows carried along these paths is the maximum. Each augmenting path $P \in \mathcal{P}$ will consume one unit of bandwidth since we deliberately assign unit capacity to the pseudo SLinks incident to S. In addition,

a path $P \in \mathcal{P}$ will contain an SNode from the set $\{f(\bar{v})|\bar{v} \in \mathcal{N}(\bar{u})\}$, since S can only be reached through these SNodes. In the event that a new path P cancels the flow on any bottleneck SLink (u, v) of any existing path $P_i \in \mathcal{P}$, Alg. 9 updates both P and P_i to exclude (u, v). Following these steps, the set of paths $\mathcal{P}$ from l to S is computed. However, the VLink re-mapping requires substrate paths from l to any SNode in $\{f(\bar{v})|\bar{v} \in \mathcal{N}(\bar{u})\}$. Hence, the algorithm identifies the path P_i that contains any SNode in $\{f(\bar{v})|\bar{v} \in \mathcal{N}(\bar{u})\}$, removes the SLink $(f(\bar{v}), S)$ from P_i, and indexes the modified P_i with the corresponding VLink $(\bar{u}, \bar{v})$. After modifying and indexing all the paths in $\mathcal{P}_l$, the algorithm returns $\mathcal{P}_l$.

To illustrate, we assume $L_1(a) = \{B, E, H\}$ in the example of Fig. 4.2. When $l = E$, Alg. 9 computes the paths carrying the maximum flow from E to S through C and G in the transformed SN of Fig. 4.2. If the augmenting path finding step in the algorithm picks the shortest path $\{EB, BG, GS\}$ as P_1, the maximum flow is restricted to 1 by the bottleneck SLink BG. Hence, the algorithm computes a new path $P = \{EH, HI, IG, GB, BA, AC, CS\}$ in the residual network as defined in the *Edmonds-Karp* algorithm. Since P cancels the flow along the bottleneck SLink $BG \in P_1$, Alg. 9 reorganizes the segments of P with P_1 to yield $\mathcal{P} = \{\{EB, BA, AC, CS\}, \{EH, HI, IG, GS\}\}$. Finally, the algorithm excludes CS and GS from the two paths and returns $\mathcal{P}_l[(a, b)] = \{EB, BA, AC\}$ and $\mathcal{P}_l[(a, c)] = \{EH, HI, IG\}$.

4.4.2 Recovery of Independent VLinks

As presented in § 4.3.5, the re-embedding of Independent VLinks in $\bar{\mathbb{E}}^f$ is a variant of the $\mathcal{NP}$-hard *Multi-commodity Unsplittable Flow Problem*. The heuristic described in § 4.4.1 is not applicable to this problem since both the endpoints of a VLink in $\bar{\mathbb{E}}^f$ are mapped to some SNodes, and finding maximum paths may yield invalid paths between the wrong pair of SNodes. Hence, we propose a greedy strategy (Alg. 10) based on computing the minimum cost path. Similar to the recovery of adjacent VLinks, Alg. 10 first sorts the VLinks in $\bar{\mathbb{E}}^f$ based on the recovery model, and $\bar{\mathscr{E}}^f$ represents this order. FRM sorts the VLinks in increasing order of their bandwidth demands, whereas PRM sorts them in decreasing order of their penalties. Such sorting orders help FRM (or, PRM) maximize (or, minimize) the number of recoveries (or, the total incurred penalties). According to this order, the algorithm computes alternate substrate path for each VLink in $(\bar{x}, \bar{y}) \in \bar{\mathscr{E}}^f$ in the subgraph induced by excluding the failed SNode and SLinks from G. For a particular VLink $(\bar{x}, \bar{y})$, the algorithm finds the minimum cost path from $f(\bar{x})$ to $f(\bar{y})$ using the procedure MCP, and adds it to the set $\mathcal{P}$. The procedure MCP uses a modified version of *Dijkstra's shortest path* algorithm [53] to take into account SLink residual capacity and VLink demand while computing the minimum cost path.

Algorithm 8: VNodes Recovery

function *VNodes Recovery*$(G, \bar{G}, V^f, E^f, recovery - model)$
  if $recovery - model = FRM$ **then**
    $\bar{\mathcal{G}} \leftarrow$ Sort $\bar{G}_i \in \bar{G}$ in increasing order of $\sum_{\forall(\bar{u},\bar{v})\in\bar{\mathcal{E}}_i^f} b_{i\bar{u}\bar{v}}$
  else if $recovery - model = PRM$ **then**
    $\bar{\mathcal{G}} \leftarrow$ Sort $\bar{G}_i \in \bar{G}$ in decreasing order of $\sum_{\forall(\bar{u},\bar{v})\in\bar{\mathcal{E}}_i^f} \pi_{i\bar{u}\bar{v}}$
  $max_{\mathcal{R}} \leftarrow 0$
  foreach $\bar{G}_i \in \bar{\mathcal{G}}$ **do**
    $\bar{u} \leftarrow$ failed virtual node in $\bar{G}_i$, $best_{\bar{u}} \leftarrow NIL$
    foreach $l \in L_i(\bar{u}) \setminus V^f$ **do**
      $\mathcal{P}_l \leftarrow Max - Paths(G, \bar{G}_i, \bar{u}, l, \bar{\mathcal{E}}_i^f, V^f, E^f)$
      if $\mathcal{R}(\mathcal{P}_l) > max_{\mathcal{R}}$ ***or*** $(\mathcal{R}(\mathcal{P}_l) = max_{\mathcal{R}}$ ***and*** $\mathcal{C}(\mathcal{P}_l) \leq \mathcal{C}(M))$ **then**
        $M \leftarrow \mathcal{P}_l$, $best_{\bar{u}} \leftarrow l$
    if $best_{\bar{u}} \neq NIL$ **then**
      map $\bar{u}$ to $best_{\bar{u}}$
      $\forall(\bar{u}, \bar{v}) \in \mathcal{E}_i^f$: map $(\bar{u}, \bar{v})$ to $M[(\bar{u}, \bar{v})]$

Algorithm 9: Max-Paths

function *Max-Paths*$(G, \bar{G}_i, \bar{u}, l, \bar{\mathcal{E}}_i^f, V^f, E^f)$
  $V \leftarrow V \setminus V^f \cup \{S\}$, $E \leftarrow E \setminus E^f$
  $\forall (\bar{u}, \bar{v}) \in \bar{\mathcal{E}}_i^f$:
    $E \leftarrow E \setminus E^f \cup \{(f(\bar{v}), S)\}$
  $max\_bw \leftarrow \max_{\forall (\bar{u},\bar{v}) \in \bar{\mathcal{E}}_i^f} \{b_{i\bar{u}\bar{v}}\}$
  $\forall (u, v) \in E$ s.t. $u = S$ or $v = S$:
    $flow_{uv} \leftarrow 0$, $c_{uv} \leftarrow 1$
  $\forall (u, v) \in E$ s.t. $r_{uv} > 0$ and $u \neq S$ and $v \neq S$:
    $flow_{uv} \leftarrow flow_{vu} \leftarrow 0$
    $c_{uv} \leftarrow c_{vu} \leftarrow \lfloor \frac{r_{uv}}{max\_bw} \rfloor$
  $G_r \leftarrow G$, $\mathcal{P} \leftarrow \phi$, $\mathcal{P}_l \leftarrow \phi$
  while $\exists$ *augmenting path* P *from* l *to* S *in* G_r **do**
    $C_f(P) \leftarrow \min_{\forall (u,v) \in P} \{c_{uv}\}$
    Augment $C_f(P)$ units flow in G_r along P
    Update residual capacity in G_r along P
    Remove links with residual capacity ≤ 0
    foreach $P_i \in \mathcal{P}$ **do**
      if P *cancels the flow along* $(u, v) \in P_i$ **then**
        $\tau \leftarrow$ Sub paths in $P_i \setminus (u, v)$
        $\nu \leftarrow$ Sub paths in $P \setminus (v, u)$
        $P \leftarrow$ Path formed by segments in τ and ν
        $P_i \leftarrow$ Path formed by segments in τ and ν
    $\mathcal{P} \leftarrow \mathcal{P} \cup P$
  $\forall P_i \in \mathcal{P}, \forall (\bar{u}, \bar{v}) \in \bar{\mathcal{E}}_i^f$:
    $P_i \leftarrow$ Path containing $(f(\bar{v}), S)$
    $\mathcal{P}_l[(\bar{u}, \bar{v})] \leftarrow P_i \setminus (f(\bar{v}), S)$
  return $\mathcal{P}_l$

Algorithm 10: VLinks-Recovery

function *VLinks-Recovery*$(G, \bar{\mathbb{E}}^f, V^f, E^f, recovery - model)$
  $\mathcal{P} \leftarrow \phi$
  $V \leftarrow V \setminus V^f$, $E \leftarrow E \setminus E^f$
  if $recovery - model = FRM$ **then**
    $\bar{\mathscr{E}}^f \leftarrow$ Sort $(\bar{x}, \bar{y}) \in \bar{\mathbb{E}}^f$ in increasing order of $b_{i\bar{x}\bar{y}}$
  else if $recovery - model = PRM$ **then**
    $\bar{\mathscr{E}}^f \leftarrow$ Sort $(\bar{x}, \bar{y}) \in \bar{\mathbb{E}}^f$ in decreasing order of $\pi_{i\bar{x}\bar{y}}$
  foreach $(\bar{x}, \bar{y}) \in \bar{\mathscr{E}}^f$ **do**
    $\mathcal{P}[(\bar{x}, \bar{y})] \leftarrow \text{MCP}(G, f(\bar{x}), f(\bar{y}), b_{i\bar{x}\bar{y}})$
    map $(\bar{x}, \bar{y})$ to $\mathcal{P}[(\bar{x}, \bar{y})]$

To illustrate, Fig. 4.2 depicts the recovery of Independent VLink *de* of VN $\bar{G}_2$, embedded on SN G according to Fig. 4.1. Alg. 10 should find alternate substrate paths between B and H for VLink *de*. The max flow based heuristic may return substrate path between B and G (or, between H and C), and is thus inappropriate. Hence, Alg. 10 recovers *de* through the minimum cost path $\{BG, GH\}$ that has sufficient bandwidth to satisfy the demand of *de*.

4.4.3 Running Time Analysis

The most expensive step in Alg. 8 is the computation of the maximum paths using Alg. 9. The core part of Alg. 9 follows the steps of *Edmonds-Karp* Algorithm. If *Edmonds-Karp* Algorithm computes augmenting paths using Breadth-first Search, it runs in $O(|V||E|^2)$ time. Since Alg. 9 is invoked $|L(\bar{u})|$ times for recovering a VN, and there are $|\bar{V}^f|$ number of affected VNs, total running time yields $O(|L(\bar{u})||\bar{V}^f||V||E|^2)$. In contrast, Alg. 10 invokes *Dijkstra's shortest path* algorithm for each VLink in $\bar{\mathbb{E}}^f$. Since *Dijkstra's shortest path* algorithm runs in $O(|E| + |V| \log |V|)$ time, Alg. 10 requires $O(|\bar{\mathbb{E}}^f|(|E| + |V| \log |V|))$ time.

4.5 Evaluation

In this section, we first present the compared approaches in § 4.5.1 followed by a description of the evaluation metrics in § 4.5.2. Then, we describe the simulation setup in § 4.5.3 and VN embedding data generation method in § 4.5.4. Finally, we present our evaluation results found through extensive simulation in § 4.5.5.

Table 4.2: Summary of Simulation Parameters

Scale	Figure	SNodes	SLinks	VNodes in a VN	VLinks in a VN	VNs	SN Utilization	VLink BW	Total Failed VLinks
Small Scale	Fig. 4.3, 4.6	50	90	5	8	10-32	20% - 75%	10% of SLink	250 - 866
	Fig. 4.4	20-65	37-118	5	8	8-19	53%	15% of SLink	178 - 519
Large Scale	Fig. 4.5, 4.7	1000	1798	3-15	2-30	93-563	20% - 80%	10% of SLink	4712 - 13546

4.5.1 Compared Algorithms

We first demonstrate the performance of FRM-based recovery by comparing *Opt-ReNoVatE* and *Fast-ReNoVatE* with an implementation of dynamic recovery [25], called *Dyn-Recovery*. For a fair comparison, we selected a related work that takes a similar approach as we did, i.e., a reactive recovery scheme that is executed centrally with a global knowledge about what are embedded on the substrate network. In addition, we exclude the last step of re-embedding the entire VN from the implementation of *Dyn-Recovery* in the event of resource inadequacy. Although we evaluate all three algorithms in small size networks, we cannot evaluate *Opt-ReNoVatE* for large networks because of the inherent complexity of ILP-solvers. Alternatively, we present *Fast-ReNoVatE*'s baseline performance assuming the SLinks of an SN have infinite bandwidth, and refer to it as *Fast-ReNoVatE-INF*. We compare Fast-ReNoVatE with Fast-ReNoVatE-INF to demonstrate the impact of residual bandwidth and the possible partitioning in a substrate network on the recovery from an SNode failure. Finally, we analyze the impact of priority-based recovery through a rigorous comparison between the two proposed recovery models *i.e.*, FRM and PRM. Since these models are orthogonal to our solutions *i.e.*, *Opt-ReNoVatE* and *Fast-ReNoVatE*, we compare the two models on each of our solutions. We differentiate between the two variants of *Opt-ReNoVatE* as *Opt-ReNoVatE-FRM* and *Opt-ReNoVatE-PRM*. Similarly, *Fast-ReNoVatE-FRM* and *Fast-ReNoVatE-PRM* represent the variants of *Fast-ReNoVatE* for FRM and PRM, respectively.

4.5.2 Performance Metrics

Recovery Efficiency

The fraction of successfully recovered VLinks over all failed VLinks expressed in percentage.

Recovery Cost

The average cost of provisioning bandwidth along a substrate path times the cost of allocating one unit bandwidth for re-embedding each failed VLink.

Execution Time

The time required for an algorithm to find the solution for all the VNs affected by an SNode failure.

Normalized Penalty

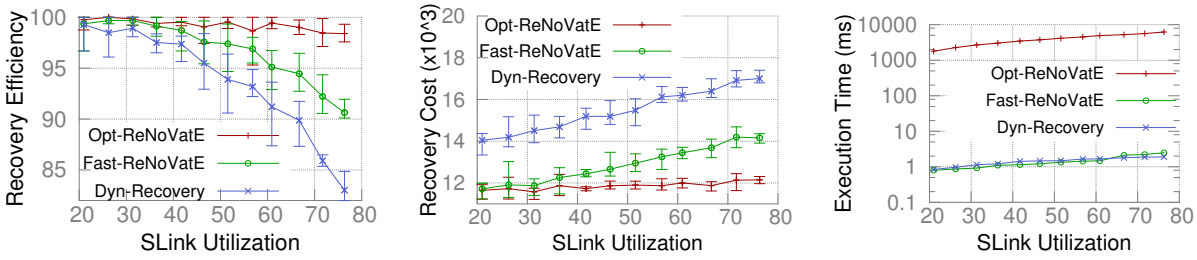
Total incurred penalty normalized with the total number of VLinks that remain unrecovered.

4.5.3 Simulation Setup

We implement *Opt-ReNoVatE* using IBM ILOG CPLEX C++ library; and *Fast-ReNoVatE* and *Dyn-Recovery* [25] using C++. We evaluate the algorithms on both small and large scale networks as summarized in Table 4.2. For each problem instance in this table, we generate 5 random SNs by taking the number of SNodes and link-to-node ratio as inputs, and randomly creating SLinks between SNodes. Then, we generate a number of VNs for each SN, and embed the VNs on the SN following the procedure described in § 4.5.4. We select a random SNode and its one hop neighbor SNodes as the location constraint set of a VNode. Afterwards, we simulate an SNode failure by removing each SNode in the SN one-by-one and execute the recovery algorithms being evaluated on the affected VNs. Finally, we measure the performance metrics of the compared algorithms by taking averages across all SNode failures for all 5 similar problem instances. The simulations are performed on a server with 2 Intel Xeon E5-2650 (8 cores @ 2.0GHz, each) processors and 256GB of RAM.

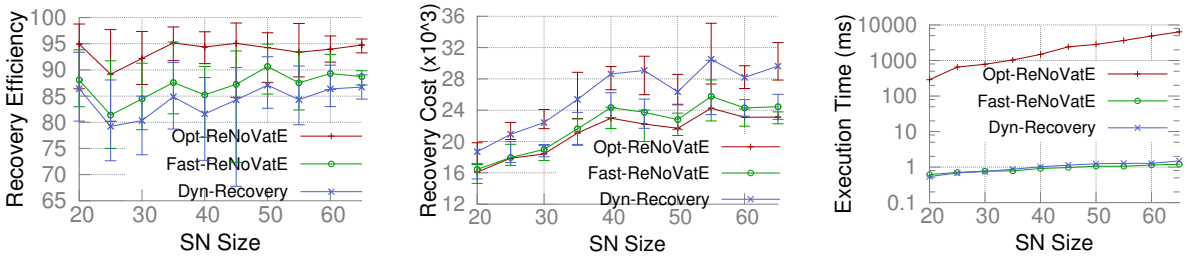
4.5.4 VN Embedding Method

VN generation and embedding are done simultaneously to overcome the issue of creating VNs that do not have feasible embeddings. The embedding of VNs on an SN is done in a random but load-balanced manner to achieve uniform distribution of VNs across the SN. Load balancing has been considered as one of the key criteria for VN embedding in previous work [59]. Achieving higher utilization (beyond 70%) of SN requires a denser embedding which is done by relaxing the load-balancing criteria. The embedding of a VN starts by randomly selecting a source SNode that has free capacity for a VNode. This forms the first VNode (source VNode) of the VN currently being embedded. For each randomly assigned neighbor of this VNode, a random destination SNode within several hop distances from the source SNode is selected as a candidate for embedding the subsequent VNode (destination VNode). The source and destination VNodes are joined using a VLink embedded on the shortest path between source and destination SNodes. If the path is not found, the destination VNode is moved to a different destination SNode. If the embedding is successful, the bandwidth demand of the VLink is generated and subtracted from the SLink residual capacity along the embedding path. To evaluate PRM, the penalty associated with the VLink is generated using a unform random distribution between 1 and the total number of failed VLinks. In the case of FRM, the penalty of each VLink is set to 1 to enforce impartial recovery. These steps are repeated until all VNodes are embedded, generating both the VN and its embedding on the SN, simultaneously.



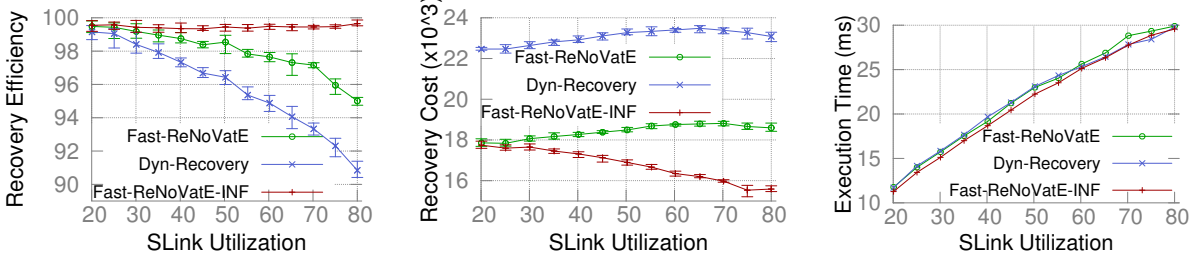
(a) Recovery Performance (b) Cost Analysis (c) Scalability Analysis

Figure 4.3: Small scale performance by varying SLink utilization



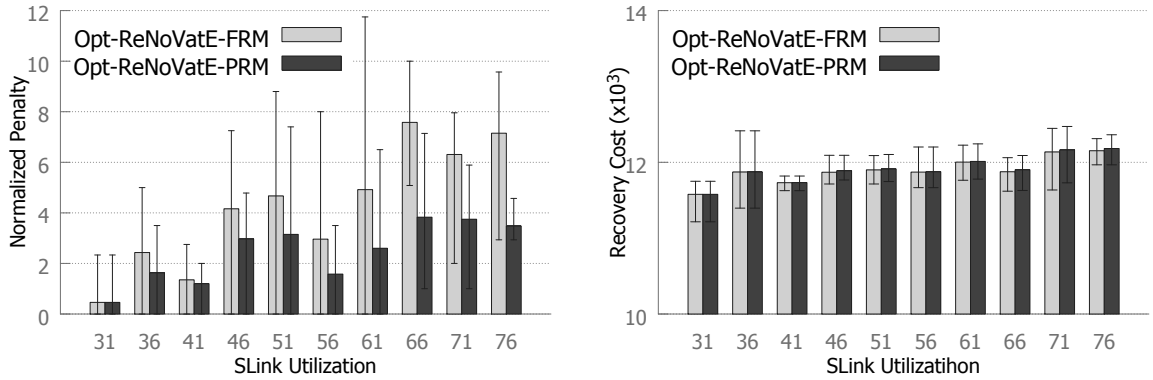
(a) Recovery Performance (b) Cost Analysis (c) Scalability Analysis

Figure 4.4: Small scale performance by varying SN size



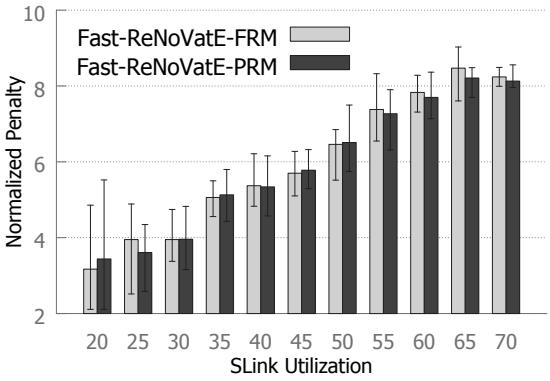
(a) Recovery Performance (b) Cost Analysis (c) Timing Performance

Figure 4.5: Performance of large scale testcases

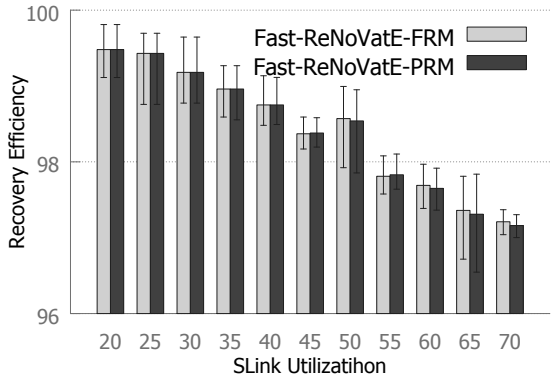


(a) Penalty Analysis (b) Cost Analysis

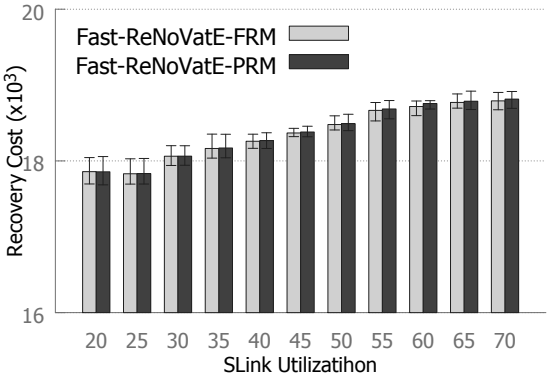
Figure 4.6: Impact of adding priority on Opt-ReNoVatE



(a) Penalty Analysis



(b) Recovery Performance



(c) Cost Analysis

Figure 4.7: Impact of adding priority on Fast-ReNoVatE

4.5.5 Results

Small Scale Evaluations

In small scale settings, we evaluate *Opt-ReNoVatE*, *Fast-ReNoVatE*, and *Dyn-Recovery* focusing on the following aspects: (i) by varying number of embedded VNs to achieve different SLink utilizations in the same SN (Fig. 4.3) (ii) by varying SN sizes while keeping the VN size and SLink utilization fixed (Fig. 4.4). We now present the impact of these aspects on our performance metrics.

Recovery Efficiency Fig. 4.3(a) shows that the recovery efficiencies of *Opt-ReNoVatE*, *Fast-ReNoVatE*, and *Dyn-Recovery* decrease with the increase in SLink utilization. As the utilization increases, more VNs are affected by the SNode failure, and less bandwidth is left for recovery resulting in the gradual decrease in the number of recovered VLinks. Further, the impact of utilization is more profound in the higher utilization cases due to the lack of load balanced embedding. In the higher utilized cases, the recovery efficiencies of *Fast-ReNoVatE* is $\sim 6\%$ better than those of *Dyn-Recovery* and $\sim 3\%$ worse than those of *Opt-ReNoVatE*. The reason behind *Fast-ReNoVatE*'s worse performance is that *Fast-ReNoVatE* recovers adjacent VLinks and independent VLinks in one particular order. In contrast, *Opt-ReNoVatE* recovers all the failed VLinks at once by exploring all possible sequences resulting in the optimal solution. Fig. 4.4(a) compares the recovery efficiencies of the three approaches for different SN sizes. As observed in the figure, recovery efficiencies of all three approaches increase slightly with the increase in SN size. This is due to the higher path diversity augmented by the higher number of SLinks in the larger SN. All the three approaches utilize the path diversity in the SN to recover more failed VLinks resulting in the increased recovery efficiencies.

Recovery Cost We show the recovery cost in Fig. 4.3(b) against different SLink utilizations. In higher utilization cases, more SLinks become saturated in terms of bandwidth, and all the algorithms have to select longer substrate paths to recover resulting in increased costs. Further, the costs of *Opt-ReNoVatE* are the least among the three, since it selects the most suitable paths through exhaustive search. In contrast, *Fast-ReNoVatE* iterates over all the affected VNs and the independent VLinks in a greedy manner, sometimes preferring less suitable paths resulting in more costs than *Opt-ReNoVatE*. Finally, *Dyn-Recovery* does not consider the cost of a path while recovering a VLink. It may select a longer path than that selected by *Fast-ReNoVatE* leading to a much larger cost than *Fast-ReNoVatE*. On average, *Fast-ReNoVatE* incurs $\sim 7\%$ more cost than *Opt-ReNoVatE* and $\sim 20\%$ less cost than *Dyn-Recovery*. Fig. 4.4(b) presents the recovery cost against different SN sizes. This figure shows that the recovery costs of all three approaches gradually increase with the increase in SN size. This is due to the fact that the two end SNodes of a failed VLink are embedded far apart from one another in a larger SN. Therefore, all the approaches recover the failed VLinks using longer substrate paths leading to higher recovery costs. However, the differences of recovery costs among the three approaches follow the same pattern as in the SLink utilization cases.

Execution Time To demonstrate the scalability of the compared algorithms, we report their execution times on different problem instances. Fig. 4.3(c) presents the execution

times by varying utilization on a fixed SN, whereas Fig. 4.4(c) presents the execution times in SNs with varying sizes while keeping the utilization fixed at $\sim$ 53%. According to Table 4.2, both problem size and the total number of failed VLinks increase with the increase in utilization and SN size. Consequently, the execution times grow for all the approaches, however, the increase is exponential for *Opt-ReNoVatE*. As it turns out, *Fast-ReNoVatE* and *Dyn-Recovery* take less than $3ms$ and $2ms$, respectively, in the largest problem instances, whereas *Opt-ReNoVatE* takes $\sim 6s$ on the same problem instances. The slightly higher execution times of *Fast-ReNoVatE* compared to *Dyn-Recovery* are due to the extra iterations of *Fast-ReNoVatE* to avoid bottleneck SLinks to achieve higher recovery efficiencies. Despite that, *Fast-ReNoVatE* is 400x$-$2000x faster than *Opt-ReNoVatE* depending on problem instance. Finally, *Opt-ReNoVatE* could not scale to more than 65 SNode SNs as shown in Fig. 4.4(c).

Large Scale Evaluations

In large scale settings, we evaluate *Fast-ReNoVatE*, *Dyn-Recovery*, and *Fast-ReNoVatE-INF* by varying SLink utilizations in the same SN(Fig. 4.5).

Recovery Efficiency Fig. 4.5(a) shows that the recovery efficiencies of *Fast-ReNoVatE* are $\sim$ 6% better than those of *Dyn-Recovery* and $\sim$ 2.5% worse than those of *Fast-ReNoVatE-INF*. Similar to the small scale results, recovery efficiencies of *Fast-ReNoVatE* and *Dyn-Recovery* decrease with the increase in SLink utilization whereas recovery efficiencies remain almost the same for *Fast-ReNoVatE-INF*. The near constant recovery efficiencies of *Fast-ReNoVatE-INF* confirms that the reason of failing to recover is the insufficiency of bandwidth in SLinks. In other words, if there were adequate bandwidth in the SLinks, *Fast-ReNoVatE* could recover $\sim$ 99% of the failed VLinks. The very small percentage of un-recovered VLinks of *Fast-ReNoVatE-INF* is due to the partitioning in the SN caused by the SNode failure. In these cases, it is not possible to recover a failed VLink even if there is sufficient bandwidth.

Recovery Cost Fig. 4.5(b) shows the recovery cost in large networks. For the same reasons discussed in the case of small scale networks, *Dyn-Recovery* incurs the largest amount of cost, and the costs of *Fast-ReNoVatE* and *Dyn-Recovery* rise with the increase in SLink utilization. In contrast, there is no effect of residual bandwidth in finding an alternate path in *Fast-ReNoVatE-INF*, and it can select the minimum cost path resulting in the least costs. This is true for *Fast-ReNoVatE* in very low utilized SNs. Further, the two

end nodes of the failed VLink are embedded closely to each other in a highly utilized SN. Hence, *Fast-ReNoVatE-INF* recovers the VLink with shorter path leading to the decrease of costs with the increase in SLink utilization. The counterintuitive behavior of Fig. 4.5(b) from SLink utilization of 70 to 80 is due to relaxing the load-balancing criteria as explained in § 4.5.4. The denser embedding to achieve higher utilization maps the VNodes of a VN closer to one another, thus requiring lower recovery cost than a sparser one.

Execution Time Fig. 4.5(c) shows that *Fast-ReNoVatE* has similar timing performance to *Dyn-Recovery*, and both are able to find a solution in less than 30ms even in the highest utilized SN.

Comparison between FRM and PRM

In this section, we compare FRM and PRM to demonstrate the impact of priority based recovery on *Opt-ReNoVatE* and *Fast-ReNoVatE*, respectively. Additionally, evaluations on *Opt-ReNoVatE* are done for small scale problem instances, whereas, *Fast-ReNoVatE* is evaluated on large scale topologies.

Impact on *Opt-ReNoVatE* Fig. 4.6 presents normalized penalties for the two variants of *Opt-ReNoVatE i.e.*, *Opt-ReNoVatE-FRM* and *Opt-ReNoVatE-PRM*. Although *Opt-ReNoVatE-FRM* and *Opt-ReNoVatE-PRM* employ two separate objective functions as presented in § 4.3, their performances in terms of recovery efficiency are found similar. Hence, we do not report those results. However, they recover different VLinks in the face of resource inadequacy. This difference can be clearly observed in their corresponding normalized penalties and recovery costs shown in Fig. 4.6(a) and Fig. 4.6(b), respectively. As seen in Fig. 4.6(a), the normalized penalties incurred by *Opt-ReNoVatE-FRM* remain always higher than those incurred by *Opt-ReNoVatE-PRM*. Even though both of them recover the same number of VLinks, *Opt-ReNoVatE-PRM* prioritizes the failed VLinks having higher penalties in order to minimize the total penalty. On the other hand, *Opt-ReNoVatE-FRM* remains indifferent to the failed VLinks while recovery leading to the higher normalized penalty. Furthermore, the normalized penalties increase with the increase in SN utilizations for both models. This is due to the higher number of VLinks being affected by an SNode failure and the lower residual capacities left for recovery in a higher utilized SN. However, *Opt-ReNoVatE-FRM* suffers more than *Opt-ReNoVatE-PRM* due to *Opt-ReNoVatE-FRM*'s impartial treatment on the failed VLinks. In contrast, *Opt-ReNoVatE-PRM* prioritizes the failed VLinks with higher penalties even though they incur

higher recovery cost. This accounts for the slightly higher recovery costs incurred by *Opt-ReNoVatE-PRM* as seen in Fig. 4.6(b).

Impact on *Fast-ReNoVatE* Fig. 4.7 demonstrates the performances of the two variants of *Fast-ReNoVatE* *i.e.*, *Fast-ReNoVatE-FRM* and *Fast-ReNoVatE-PRM* in terms of our evaluation metrics. Fig. 4.7(a) shows the normalized penalties incurred by the two models by varying SN utilizations in large scale topologies. The graph of Fig. 4.7(a) follows similar trend as we observe in the case of *Opt-ReNoVatE* in Fig. 4.6 and reinforces the argument presented in the previous paragraph. However, the deviations between *Fast-ReNoVatE-FRM* and *Fast-ReNoVatE-PRM* are less profound than those of the models of *Opt-ReNoVatE*. The reason behind such difference is that *Fast-ReNoVatE-PRM* employs priority on the VN level rather than on the VLink level as employed by *Opt-ReNoVatE-PRM*. Unlike the models of *Opt-ReNoVatE*, *Fast-ReNoVatE-FRM* and *Fast-ReNoVatE-PRM* show slight differences in terms of recovery efficiency and cost. Therefore, we present those results in Fig. 4.7(b) and Fig. 4.7(c). According to these figures, the recovery efficiencies (or, costs) of *Fast-ReNoVatE-PRM* are slightly lower (or, higher) than those of *Fast-ReNoVatE-FRM*. This stems from the fact that *Fast-ReNoVatE-PRM* prioritizes the VLinks with higher penalties despite being recovered in longer substrate paths. Such recovery increases the recovery cost and reduces the residual capacity of the SLinks leaving less room for re-embedding some VLinks. This accounts for the higher recovery costs and lower recovery efficiencies of *Fast-ReNoVatE-PRM* than those of *Fast-ReNoVatE-FRM*.

Impact on Execution Time Since FRM and PRM are orthogonal to our solutions *Opt-ReNoVatE* and *Fast-ReNoVatE*, incorporating the models to our solutions does not bring any significant change. Hence, the complexity of *Opt-ReNoVatE* presented in 4.3.5 and the running time analysis of *Fast-ReNoVatE* presented in 4.4.3 remain the same irrespective of the model being chosen. In our simulations, the execution times of *Opt-ReNoVatE-PRM* and *Fast-ReNoVatE-PRM* are found similar to their FRM counterparts. Hence, we refrain from reporting them in this chapter.

4.6 Conclusion

In this chapter, we have addressed the problem of generalized recovery of a batch of affected VNs, resulting from a single substrate node failure. We have formulated the problem as an Integer Linear Programming (ILP) model, *Opt-ReNoVatE* and presented an efficient

heuristic algorithm, *Fast-ReNoVatE*, to tackle the computational complexity. We have evaluated *Fast-ReNoVatE*, *Opt-ReNoVatE*, and a state-of-the-art solution, *Dyn-Recovery* in both small and large scale networks. Evaluation results demonstrate that *Fast-ReNoVatE* can recover $\sim$ 6% more VLinks than *Dyn-Recovery* and $\sim$ 3% less VLinks than *Opt-ReNoVatE* in high utilization scenarios. In terms of scalability, *Fast-ReNoVatE* is several orders of magnitude faster than *Opt-ReNoVatE*, and has comparable performance with *Dyn-Recovery*. In large scale networks, we have compared *Fast-ReNoVatE* with *Dyn-Recovery* and a baseline case of infinite bandwidth SN. These results demonstrate that *Fast-ReNoVatE* is able to recover $\sim$ 99% of the failed VNs if the SN has adequate residual capacity, and has similar timing performance to *Dyn-Recovery*. Furthermore, we have investigated two variants of our recovery scheme, namely, fair recovery model (FRM) and priority-based recovery model (PRM). Our evaluation results suggest that FRM-based solutions fail to take into account variety of recovery requirements. In contrast, PRM-based solutions can prioritize the affected VNs based on SLA requirements, impacts of failure or profits, and adhere to that priority during recovery.

Chapter 5

Reliable Slicing of Elastic Optical Networks

Transport networks employing the latest advances in elastic optical networking will form the backbone of the fifth-generation (5G) networks and beyond [68, 12, 28]. Transport-network capacity must be easily partitionable to facilitate network slicing for 5G services such as enhanced mobile broadband and ultra-reliable low latency communications [60]. Elastic Optical Networks (EONs) are an excellent choice for establishing transport network slices, thanks to their ability to tailor network resources based on service requirements [73].

Network slices are usually provided in the form of virtual networks (VNs) (*i.e.*, collection of virtual nodes and virtual links (VLinks)). These VNs are anticipated to host services operating at hundreds of Gbps line-rate. As a consequence, even a short-lived network outage can cause a significant traffic disruption in these slices. However, ensuring reliability of network slices is non-trivial as failures in a transport network, such as transponder failure and fiber-cuts, are still the norm. A viable solution to ensure slice reliability is to pre-provision dedicated backup paths for each path used to embed the VLinks, also known as dedicated protection [95]. Dedicated backup paths can allow fast fail-over switching within milliseconds [9], but they incur a significant resource overhead since they remain idle during failure-free operations

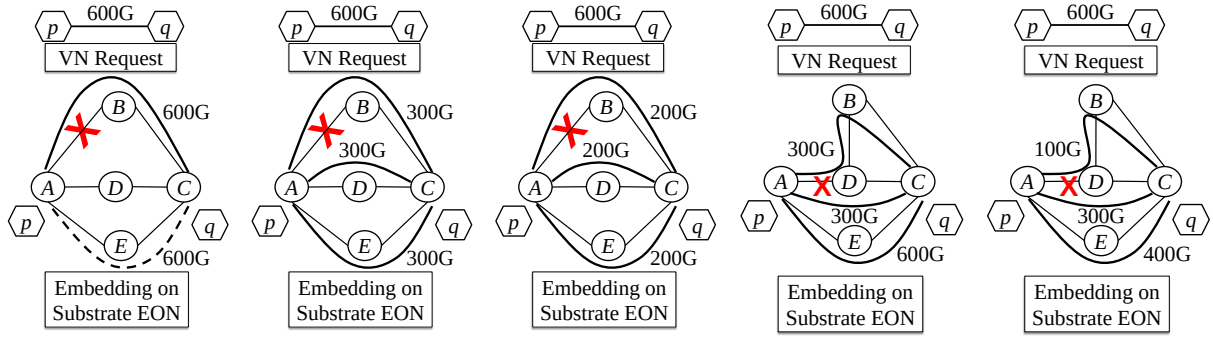
In this chapter, we propose a VN embedding solution with dedicated protection that incorporates two techniques to decrease the network-resource consumption: (i) bandwidth squeezing, *i.e.*, the opportunity to tune the amount of bandwidth guaranteed in case of failures [136, 151, 42, 16]; and (ii) survivable multi-path provisioning [83, 93], *i.e.*, VLink demand splitting over multiple disjoint paths. Bandwidth squeezing is motivated by the

fact that not all services require the full bandwidth to be operational, *e.g.*, a file transfer service can still be operational with reduced bandwidth during a failure. The extent of bandwidth reduction is measured by bandwidth squeezing rate (BSR), which is the percentage of original bandwidth made available after a failure. Survivable multi-path provisioning allows us to exploit the fact that not all of the disjoint splits belonging to the same VLink are jointly affected by a single substrate link failure. Therefore, the bandwidth of the surviving paths after a failure can be reused to provide a given BSR. Note that, in this chapter, we consider a single substrate link failure scenario, which is more common than multiple simultaneous link failures [117].

The illustrative example in Fig. 5.1 demonstrates how the synergy of bandwidth squeezing and survivable multi-path provisioning can reduce backup resources for dedicated protection. If we do not allow any splitting of a VLink demand (*i.e.*, in our example, a 600Gbps demand is carried over a single primary path *A*-*B*-*C*), then we require 2× the original bandwidth for providing a dedicated protection path (*A*-*E*-*C*), as illustrated in Fig. 5.1(a). However, if we split the demand over three disjoint paths as in Fig. 5.1(b), then any two of the three paths will be active after a single substrate link failure. In this way 100% bandwidth will still be available after a failure, even though we are allocating a total bandwidth that is only 1.5× the original demand. Bandwidth savings can even be higher if we employ bandwidth squeezing, *e.g.*, Fig. 5.1(c) shows how 66% BSR is ensured without any additional backup bandwidth. However, the advantages of multi-path provisioning depend on the availability of disjoint paths in the substrate network. For instance, the lack of sufficient number of disjoint paths reduces the gain of multi-path provisioning regardless of the extent of BSR as shown in Fig. 5.1(e) and Fig. 5.1(d).

In this chapter, we present our solutions for reliable slicing of 5G transport networks based on VN embedding with dedicated protection over an EON. Our solutions leverage EON's capabilities in finer-grained spectrum allocation, adapting modulation format and Forward Error Correction (FEC) overhead for rightsize resource allocation to the VNs. We perform simulations using realistic network topologies, which provide valuable insight into how different levels of BSR and path diversity in the substrate EON impact the extent of backup resource savings for dedicated protection. We also analyze the steady state behavior of our heuristic solution using a discrete event simulator. For instance, our evaluation shows that by using multi-path provisioning it is possible to guarantee up to 40% of the requested bandwidth of a VN during failure (*i.e.*, $BSR \leq 40\%$) while using as low as 10% additional spectrum resources. Consequently, VN blocking ratio for $BSR \leq 40\%$ remains very similar to that of the case with no backup.

The rest of the chapter is organized as follows. We discuss the relevant research literature in § 5.1 and problem statement in § 5.2. Then, we formulate the optimal solution as



(a) No virtual link demand splitting (100% BSR)

(b) Splitting over disjoint paths (100% BSR)

(c) Splitting over disjoint paths (66% BSR)

(d) Splitting over non-disjoint paths (100% BSR)

(e) Splitting over non-disjoint paths (66% BSR)

Figure 5.1: Benefits of virtual link demand splitting and the impact of availability of disjoint paths

an Integer Linear Program (ILP) in § 5.3 followed by a heuristic algorithm to solve large problem instances in § 5.4. In § 5.5, we discuss the numerical evaluation of the proposed approaches over realistic network instances. Finally, we present our concluding remarks in § 5.6.

5.1 Related Works

Reliability aspects of VN embedding [132] and, more generally, of traffic provisioning in fixed-grid optical networks [134] have been extensively studied. Techniques to enhance reliability [134] are typically categorized in two broad classes: protection and restoration. The first class contains proactive approaches where backup paths are provisioned at the time a VN or a traffic matrix is provisioned, which leads to faster recovery time. Backup paths can be either dedicated [173, 48] or shared among multiple requests [132, 144, 71]. In contrast, restoration techniques provision backup resources after a failure has occurred [143]. Restoration alleviates the issue of keeping a large amount of backup resources idle during regular operation, however it has significantly higher recovery time, not suitable for supporting ultra-reliable communications in 5G networks [134, 35], and will not be considered in this study. Similarly, shared protection could be used to reduce the amount of backup resources of dedicated protection [111, 149, 172, 167, 30, 164, 175], but at the cost of longer protections switching time [126, 75], hence in our study we focus on dedicated protection.

In the following, we discuss the state-of-the-art research literature on VNE with dedicated protection and the literature on backup resource optimization techniques for VNE.

5.1.1 VNE with Dedicated Protection

Optical backbone networks have traditionally had very strict requirement on failure recovery time, typically in the order of few tens of milliseconds [134]. Such strict recovery time will continue to be the norm in 5G transport networks as well [9]. Fast failover within such short time can be achieved only through dedicated protection. Dedicated protection has its origin from survivable WDM optical networks where lightpaths are established with a dedicated backup path for quick recovery from fiber cuts [134]. In the context of 5G transport network slicing, dedicated protection can be in the form of providing dedicated protection to each virtual link of a network slice or VN. The concept of dedicated protection can be further extended to a whole VN topology as proposed in the research literature [165]. Ye *et al.*, address the dedicated VN topology protection problem by proposing a quadratic integer program and a greedy heuristic [173]. Later, Chowdhury *et al.*, modeled the VN topology protection problem as a variation of graph partitioning problem and proposed an ILP formulation to solve it optimally [48].

A weaker form of dedicated topology protection problem has been studied by Jiang *et al.*, [88]. The authors' proposed a scheme that embeds the backup virtual nodes disjointly from the primary virtual node embedding. However, the backup nodes can share substrate nodes among themselves, in this way reducing the degree of survivability. Also, they do not provision full bandwidth for the backup paths, rather, provision some backup paths in advance that can be used by the virtual links during failure.

The classical routing and spectrum allocation problem in EON with dedicated path protection has been addressed in [98, 99, 163]. However, the problem of reliable VNE over EON has gained attention only recently. Some research works have addressed the VN embedding problem over EON with dedicated path protection [170, 39, 180] while minimizing spectrum occupation, number of regenerators [170, 39] or energy consumption [180]. These works formulated the problem using a path-based ILP, where a disjoint backup path for each of the usable primary paths is pre-computed. Furthermore, they assume the modulation format is already selected for each bitrate (*e.g.*, 100Gbps bitrate is provisioned using only DP-QPSK modulation format), further limiting the solution space. In contrast, [96] proposed a modulation-adaptive link-disjoint path selection model by considering a step function based on realistic modulation formats in order to minimize the total number of utilized spectrum slots for dedicated protection.

Another protection technique for EONs is the use of *p-cycles* [169, 87, 41]. A p-cycle is a pre-configured cycle that act as a backup path for a link in the working paths of an EON. Unlike dedicated protection, backup p-cycles are needed for each EON link and recovery is performed locally in each EON link instead of providing end-to-end protection. As such backup spectrum requirement can be very high unless backup resources are shared among multiple working paths that, in consequence, may exceed the recovery time required by 5G transport network services. In this chapter, we consider multi-path provisioning, bandwidth squeezing, and capture all the tunable transmission parameters made available by EONs, which is not considered by the existing works in the literature.

5.1.2 Backup resource optimization techniques for VNE

To save network resources, a number of studies considered a alternate form of protection, called bandwidth squeezing, where only a reduced bandwidth is guaranteed after a failure, according to a given BSR [136, 151, 149, 42, 16, 66]. The advantages of using bandwidth squeezing can be amplified when combined with virtual link demand splitting over multiple disjoint substrate paths [136, 16, 137]. The latter is commonly known as survivable multi-path provisioning [83, 137, 138, 66, 93]. Majority of the survivable multi-path approaches, except [66], assume that all the paths used for embedding a virtual link are link disjoint, and may not explore the complete solution space. In addition, the authors in [34] proposed a scheme that consists in ensuring part of the minimum bitrate by protection as per BSR and complementing the rest by restoration. In contrast to these existing works, our work stands out by considering a VN topology embedding and by allowing splitting of a virtual link demand not only over non-disjoint paths but also over multiple spectral segments on the same path, significantly increasing the complexity of the problem as discussed in § 5.3 and 5.4.

5.2 Mathematical Model and Problem Statement

We first present a mathematical model of the inputs followed by a formal statement of the problem. Without loss of generality, we assume the optical nodes to be colorless, directionless, and contentionless [14].

5.2.1 Substrate EON

The substrate EON (SN) is an undirected graph $G = (V, E)$, where V and E are the set of substrate optical nodes (SNodes) and substrate optical links (SLinks), respectively. We assume the SLinks to be bi-directional, *i.e.*, adjacent optical nodes are connected by one optical fiber in each direction. The optical frequency spectrum on each SLink $e = (u, v) \in E$ is divided into equal-width frequency slots represented by the set S and enumerated as $1, 2 \ldots |S|$. $\mathcal{P}$ and $\mathcal{P}^k_{uv} \subset \mathcal{P}$ represent the set of all paths in G and the set of k-shortest paths between nodes $u, v \in V$, respectively. The number of SLinks and the physical length of a path p in kilometers are represented by $|p|$ and $len(p)$, respectively. We use the binary variable δ_{pe} to denote the association between a path $p \in \mathcal{P}$ and an SLink $e \in E$.

The following transmission parameters can be configured on a path p with length $len(p)$ to enable data transmission with different data-rates $d \in \mathcal{D}$: *baud-rate* or *symbol-rate*, b, *modulation format*, m, and *FEC overhead*, f, selected from the set of possible values $\mathcal{B}$, $\mathcal{M}$, and $\mathcal{F}$, respectively. We use a tuple $t = (d, b, m, f) \in \mathcal{T} = (\mathcal{D} \times \mathcal{B} \times \mathcal{M} \times \mathcal{F})$ to represent a transmission configuration that dictates the combination of $b \in \mathcal{B}$, $m \in \mathcal{M}$, and $f \in \mathcal{F}$ yielding a data-rate $d \in \mathcal{D}$. We use $t^{(d)}, t^{(b)}, t^{(m)}$, and $t^{(f)}$ to denote the data-rate, baud-rate, modulation format, and FEC overhead of a configuration $t \in \mathcal{T}$. A *reach table* $\mathcal{R}$, computed based on physical layer characteristics, specifies the maximum length of a path (*i.e.*, the *reach* r_t) capable of retaining a satisfactory optical signal to noise ratio when configured according to a transmission configuration $t \in \mathcal{T}$. Finally, n_t denotes the number of slots required for a transmission configuration $t \in \mathcal{T}$, which is dependent on the parameters of t.

5.2.2 Virtual Network

The virtual network (VN) is represented by an undirected graph $\bar{G} = (\bar{V}, \bar{E})$, where $\bar{V}$ and $\bar{E}$ are the set of virtual nodes (VNodes) and virtual links (VLinks), respectively. The function $\tau : \bar{V} \rightarrow V$ represents VNode to SNode mapping and is an input to our problem (a common assumption for optical network virtualization [178]). Each virtual link $\bar{e} \in \bar{E}$ has a bandwidth requirement $\bar{\beta}_{\bar{e}}$ and a reliability requirement $0 < BSR_{\bar{e}} \leq 100$, which indicates the percentage of original bandwidth that should be available after an SLink fails. $BSR_{\bar{e}}$ can also be used to realize a bandwidth profile with a maximum and minimum demand similar to [73]. We allow VLinks to be mapped on multiple substrate paths (SPaths) (similar to [127, 147]), each with a lower data-rate than $\bar{\beta}_{\bar{e}}$. Splitting $\bar{\beta}_{\bar{e}}$ over multiple SPaths is a feasible way to support higher data-rates (*e.g.*, $\geq$ 400Gbps) that

limit the number of usable paths due to their shorter reaches. However, we restrict the number of VLink splits to maximum q (≥ 1). Such multi-path embedding is supported by technologies such as Virtual Concatenation (VCAT) in Optical Transport Network (OTN) [24] or bonding capabilities of FlexEthernet [8].

5.2.3 Problem Statement

Given an SN G, a reach table $\mathcal{R}$, and a VN request $\bar{G}$ with given VNode mapping function τ:

- Compute the link embedding function $\gamma : \bar{E} \rightarrow \chi : \chi \subset \mathcal{P} \times \mathcal{T} \times S^2$ and $1 \leq |\chi| \leq q$, *i.e.*, compute up to a maximum of q splits for each VLink $\bar{e} \in \bar{E}$ such that $0.01 \times BSR_{\bar{e}} \times \bar{\beta}_{\bar{e}}$ bandwidth is available during an SLink failure and at least $\bar{\beta}_{\bar{e}}$ bandwidth is available during the rest of the time. For each split, γ should select an SPath and an appropriate transmission configuration $t \in \mathcal{T}$ from the reach table $\mathcal{R}$, and allocate a contiguous segment of slots represented by the starting and ending slot index on each SLink along the SPath. Note that the same SPath can be used multiple times as the splits of a VLink following the reasoning in [147, 155]. $\chi_{\bar{e}i} = (p, t, s_b, s_t) | 1 \leq i \leq q$ represents the i-th split, where $\chi^{(p)}_{\bar{e}i}$ and $\chi^{(t)}_{\bar{e}i}$ denote the selected SPath and transmission configuration for the i-th split, respectively. In addition, allocation of spectrum slots for the i-th split begins at index $\chi^{(s_b)}_{\bar{e}i}$ and ends at index $\chi^{(s_t)}_{\bar{e}i}$ along each SLink in the SPath $\chi^{(p)}_{\bar{e}i}$.
- The total number of slots required to provision the VN is minimum according to the following cost function:

$$\sum_{\forall \bar{e} \in \bar{E}} \sum_{i=1}^{q} (\chi^{(s_t)}_{\bar{e}i} - \chi^{(s_b)}_{\bar{e}i} + 1) \times |\chi^{(p)}_{\bar{e}i}| \tag{5.1}$$

  Here, $|\chi^{(p)}_{\bar{e}i}|$ is the number of SLinks on the SPath $\chi^{(p)}_{\bar{e}i}$.

The above is subject to substrate resource constraints, and spectral contiguity (*i.e.*, the allocated slots of each split are always adjacent to each other) and continuity (*i.e.*, the same sequence of slots are allocated on each SLink along an SPath) constraints on the lightpaths.

5.2.4 Pre-computations

For each VLink $\bar{e} \in \bar{E}$, we pre-compute $\mathcal{P}_{\bar{e}}^{k}$, a set of k shortest paths between the pair of SNodes where the VLink's endpoints' are mapped. For each SPath $p \in \mathcal{P}_{\bar{e}}^{k}$, we pre-compute the set of admissible transmission configurations, $\mathcal{T}_{\bar{e}p} \subset \mathcal{T}$, such that each configuration $t \in \mathcal{T}_{\bar{e}p}$ results in a reach $r_t \geq len(p)$ and has a data-rate $t^{(d)}$. $\mathcal{T}_{\bar{e}}$ contains all the distinct tuples suitable for $\bar{e}$ and is defined as $\bigcup_{\forall p \in \mathcal{P}_{\bar{e}}^{k}} \mathcal{T}_{\bar{e}p}$.

5.3 Problem Formulation

We present a path-based ILP formulation for optimally solving our problem. Note that some of the constraints, except the reliability constraints, have been presented in different forms in different research works [166, 44, 147]. In the interest of completeness, we report all the constraints in the following. Table 5.1 presents a glossary of key notation used in the ILP formulation.

5.3.1 Decision Variables

We allow a VLink's bandwidth demand to be satisfied by provisioning spectrum slots over one or more SPaths where an SPath can be used more than once (up to a maximum of q) as discussed in 5.2.3. To model the same SPath appearing more than once in a VLink's embedding, we assume each transmission configuration on an SPath can be instantiated multiple times (up to a maximum of q times). The following variable represents VLink mapping:

$$w_{\bar{e}pti} = \begin{cases} 1 & \text{if } \bar{e} \in \bar{E} \text{ uses } i\text{-th instance of } t \in \mathcal{T}_{\bar{e}p} \\ & \text{ on path } p \in \mathcal{P}_{\bar{e}}^{k} \\ 0 & \text{otherwise} \end{cases}$$

Finally, the following decision variable creates the relationship between a mapped SPath and the slots in its SLinks:

$$y_{\bar{e}ptis} = \begin{cases} 1 & \text{if } \bar{e} \in \bar{E} \text{ uses slot } s \in S \text{ on path } p \in \mathcal{P}_{\bar{e}}^{k} \\ & \text{with the } i\text{-th instance of } t \in \mathcal{T}_{\bar{e}p} \\ 0 & \text{otherwise} \end{cases}$$

Table 5.1: Notation Table

Inputs & Pre-computations	
$G = (V, E)$	Substrate EON
$\bar{G} = (\bar{V}, \bar{E})$	Virtual Network (VN)
S	The set of equal-width frequency slots
$\mathcal{P}$	Set of all paths in G
$\mathcal{P}_{uv}^{k} \subset \mathcal{P}$	k-shortest paths between u and v in G
$\mathcal{D}, \mathcal{B}, \mathcal{M}, \mathcal{F}$	Set of all possible data-rates, baud-rates, modulation formats, & FEC overhead levels
$\mathcal{T} = (\mathcal{D} \times \mathcal{B} \times \mathcal{M} \times \mathcal{F})$	Set of all possible transmission configurations
$t = (d, b, m, f) \in \mathcal{T}$	A transmission configuration *i.e.*, a baud-rate b, modulation format m, and FEC overhead f yielding data-rate d
$t^{(d)} \in \mathcal{D}, t^{(b)} \in \mathcal{B}$, $t^{(m)} \in \mathcal{M}, t^{(f)} \in \mathcal{F}$	data-rate, baud-rate, modulation format, and FEC overhead of transmission configuration t
$\mathcal{R}$	Reach table
$r_t \in \mathcal{R}$	Achievable reach for configuration t
n_t	No. of slices required for configuration t
$\mathcal{P}_{\bar{e}}^{k}$	k candidate shortest paths for VLink $\bar{e} \in \bar{E}$
$\mathcal{T}^{\bar{e}p} \subset \mathcal{T}$	Admissible transmission configurations on path p for embedding VLink $\bar{e} \in \bar{E}$
Decision Variables	
$w_{\bar{e}pti}$	Indicates if $\bar{e} \in \bar{E}$ uses i-th instance of transmission configuration $t \in \mathcal{T}_{\bar{e}p}$ on $p \in \mathcal{P}_{\bar{e}}^{k}$
$y_{\bar{e}ptis}$	Indicates if $\bar{e} \in \bar{E}$ uses slot $s \in S$ on $p \in \mathcal{P}_{\bar{e}}^{k}$ with the i-th instance of transmission configuration $t \in \mathcal{T}_{\bar{e}p}$

5.3.2 Constraints

VLink Demand Constraints

We provision a VLink by splitting it across multiple (up to q) SPaths. Each of these splits are then assigned a transmission configuration, resulting in provisioning a data-rate along the split. Constraint (5.2) ensures that for each VLink $\bar{e} \in \bar{E}$, the sum of data-rates resulting from applying the selected transmission configuration on the selected splits is equal or larger than the VLink's demand. (5.3) enforces an upper limit on the number of splits in order to keep splitting overhead within a operator defined threshold.

$$\forall \bar{e} \in \bar{E} : \bar{\beta}_{\bar{e}} \leq \sum_{\forall p \in \mathcal{P}_{\bar{e}}^{k}} \sum_{\forall t \in \mathcal{T}_{\bar{e}p}} \sum_{i=1}^{q} (w_{\bar{e}pti} \times t^{(d)}) \tag{5.2}$$

$$\forall \bar{e} \in \bar{E} : \sum_{\forall p \in \mathcal{P}_{\bar{e}}^{k}} \sum_{\forall t \in \mathcal{T}_{\bar{e}p}} \sum_{i=1}^{q} w_{\bar{e}pti} \leq q \tag{5.3}$$

Slot Assignment and Spectral Contiguity constraints

We ensure by (5.4) that if a path p is selected with a specific transmission configuration t, then the required number of slots n_t to support the data-rate $t^{(d)}$ is allocated on the path. Then, (5.5) ensures that each slot on an SLink is allocated to at most one path. Finally, (5.6) enforces spectrum contiguity constraint [147], *i.e.*, ensures the slots allocated on each SLink of a path form a contiguous frequency spectrum.

$$\forall \bar{e} \in \bar{E}, \forall p \in \mathcal{P}_{\bar{e}}^{k}, \forall t \in \mathcal{T}_{\bar{e}p}, 1 \leq i \leq q : \sum_{\forall s \in S} y_{\bar{e}ptis} = n_t w_{\bar{e}pti} \tag{5.4}$$

$$\forall e \in E, \forall s \in S : \sum_{\forall \bar{e} \in \bar{E}} \sum_{\forall p \in \mathcal{P}_{\bar{e}}^{k}} \sum_{\forall t \in \mathcal{T}_{\bar{e}p}} \sum_{i=1}^{q} w_{\bar{e}pti} y_{\bar{e}ptis} \delta_{pe} \leq 1 \tag{5.5}$$

$$\forall \bar{e} \in \bar{E}, \forall p \in \mathcal{P}_{\bar{e}}^{k}, \forall t \in \mathcal{T}_{\bar{e}p}, 1 \leq i \leq q, 1 \leq s \leq |S| - 1 :$$
$$\sum_{s'=s+2}^{|S|} y_{\bar{e}ptis'} \leq |S| \times (1 - y_{\bar{e}ptis} + y_{\bar{e}pti(s+1)}) \tag{5.6}$$

Survivability Constraints

We must ensure that for each single substrate link failure scenario, the sum of data-rates of the unaffected splits of a VLink $\bar{e} \in \bar{E}$ is at least $BSR_{\bar{e}}$ percentage of the original VLink demand. This is enforced by (5.7) as follows:

$$\forall \bar{e} \in \bar{E}, \forall e \in E : (\sum_{\forall p \in \mathcal{P}_{\bar{e}}^{k}} \sum_{\forall t \in \mathcal{T}_{\bar{e}p}} \sum_{i=1}^{q} w_{\bar{e}pti} t^{(d)})$$
$$-(\sum_{\forall p \in \mathcal{P}_{\bar{e}}^{k}} \sum_{\forall t \in \mathcal{T}_{\bar{e}p}} \sum_{i=1}^{q} w_{\bar{e}pti} t^{(d)} \delta_{pe}) \geq 0.01 BSR_{\bar{e}} \bar{\beta}_{\bar{e}} \tag{5.7}$$

5.3.3 Objective Function

Our cost function minimizes the total number of spectrum slots required to embed all the VLinks of a VN as shown in the first part of (5.8). However, to break ties among multiple solutions with the same total number of slots, we use the second term with a fractional weight ϵ in (5.8). This term minimizes the number of splits over all the VLinks since each split requires a pair of transponders at the source and destination node of a lightpath. This gives us the following objective function:

$$\text{minimize}(\sum_{\forall \bar{e} \in \bar{E}} \sum_{\forall p \in \mathcal{P}_{\bar{e}}^{k}} \sum_{\forall t \in \mathcal{T}_{\bar{e}p}} \sum_{i=1}^{q} \sum_{\forall s \in S} y_{\bar{e}ptis} |p| +$$
$$\epsilon \sum_{\forall \bar{e} \in \bar{E}} \sum_{\forall p \in \mathcal{P}_{\bar{e}}^{k}} \sum_{\forall t \in \mathcal{T}_{\bar{e}p}} \sum_{i=1}^{q} w_{\bar{e}pti}) \tag{5.8}$$

5.3.4 Complexity of the ILP

The problem of VN embedding on EON with dedicated path protection and multi-path provisioning is reducible to routing and spectrum allocation in EON, which is known to be NP-hard [38]. This also makes the problem addressed in this chapter NP-hard. The complexity of an ILP solution for optimally solving a problem will depend on the number of variables and constraints in the formulation. In case of our formulation, it will require enumerating $O(|\bar{E}|\mathcal{P}\mathcal{T}qS)$ variables while considering $O(|S||E| + |S||\bar{E}|\mathcal{P}\mathcal{T}q)$ constraints.

5.4 Heuristic Algorithm

Given the limited scalability of the ILP formulation, we develop a heuristic algorithm to solve large instances of our problem. In this section, we first give an overview of the main steps involved in reliable VN embedding for a given node mapping (§ 5.4.1). Then, we discuss how we select the order of VLinks to be embedded for increasing the chances of finding a feasible solution (§ 5.4.2). Finally, we discuss the embedding process of a single VLink (§ 5.4.3).

5.4.1 Heuristic Solution for Reliable VN Embedding

Algorithm 11: Compute VN Embedding

1 **function** *EmbedVN*($G, \bar{G}, \tau$)
2 $\quad \bar{\mathcal{E}} \leftarrow$ GetVLinkOrder($G, \bar{G}$)
3 $\quad$ **foreach** $\bar{e} \in \bar{\mathcal{E}}$ *in the sorted order* **do**
4 $\quad\quad \mathcal{I}_{\bar{e}} \leftarrow$FindEmbedding($G, \bar{e}, \mathcal{P}^i_{\bar{e}}, \mathcal{T}_{\bar{e}}$)
5 $\quad\quad$ **foreach** $e \in p | p \in \mathcal{I}_{\bar{e}}.\mathbb{P}$ **do**
6 $\quad\quad\quad$ Perform slot assignment using $\mathcal{I}_{\bar{e}}.\mathcal{S}$
7 $\quad\quad \chi_{\bar{e}}.\mathcal{P} \leftarrow \mathcal{I}_{\bar{e}}.\mathbb{P}$, $\chi_{\bar{e}}.\mathcal{T} \leftarrow \mathcal{I}_{\bar{e}}.\mathbb{T}_{\mathbb{P}}$
8 $\quad\quad$ **if** $\chi_{\bar{e}} = \phi$ **then**
9 $\quad\quad\quad$ **return** $< \phi, \phi >$
10 $\quad$ **return** $< \tau : \bar{V} \rightarrow V, \gamma : \bar{E} \rightarrow \chi >$

Alg. 11 takes as input a VN $\bar{G}$, an EON G, and a node mapping function $\tau : \bar{V} \rightarrow V$. One way of computing a cost efficient VLink mapping for a given VNode mapping is to consider all $|\bar{E}|!$ possible orders for sequentially embedding VLinks. The lowest cost mapping among all possible VLink orders then can be chosen as a cost effective solution for the given VNode mapping. However, this brute-force approach is not scalable. Instead, Alg. 11 considers only one sequential VLink order that is computed to converge to a solution within a reasonable time. To increase the chances of finding a feasible VN embedding, Alg. 11 invokes Alg. 12 that finds a good order of the VLinks ($\bar{\mathcal{E}}$) to embed. For each VLink $\bar{e} \in \bar{\mathcal{E}}$ in the computed order, Alg. 11 finds the embedding solution based on $\mathcal{P}^i_{\bar{e}}$ by invoking Alg. 13 (line 4). Alg. 11 then allocates spectrum slots on all SLinks present in the solution and updates the VLink embedding $\chi_{\bar{e}}$ accordingly. If no such solution for $\bar{e}$ can be found, the embedding of the VN is rejected.

5.4.2 Compute VLink Ordering

Algorithm 12: Find a VLink embedding order

function *GetVLinkOrder*$(G, \bar{G})$
  $Z = (N, A) \leftarrow \text{AuxGraph}(\bar{G})$, $O[1...n] \leftarrow \phi$, $i \leftarrow n$
  while $N \neq \phi$ **do**
    foreach $n_{\bar{e}_i} \in N$ **do**
      Compute $w(n_{\bar{e}_i})$ using (5.12)
    $n^{min}_{\bar{e}_i} \leftarrow n_{\bar{e}_i}$ with minimum $w(n_{\bar{e}_i})$
    $O[i] \leftarrow \bar{e}_i$ corresponding to $n^{min}_{\bar{e}_i}$
    $N \leftarrow N - \{n_{\bar{e}_i}\}$
    foreach $(n^{min}_{\bar{e}_i}, n_{\bar{e}_j}) \in A$ **do** $A \leftarrow A \setminus \{(n^{min}_{\bar{e}_i}, n_{\bar{e}_j})\}$
    $i \leftarrow i - 1$
  return O

Alg. 12 finds an order of VLinks, O, that increases the chances of finding a feasible embedding for all the VLinks. Computing a feasible embedding of a VLink depends on the availability of contiguous slots in the candidate SPaths of that VLink. If the candidate SPaths $\mathcal{P}^k_{\bar{e}_i}$ of a VLink $\bar{e}_i$ have many SLinks that are common with the candidate SPaths of other VLinks that appear before $\bar{e}_i$ in a VLink order o, the slots of these SLinks (correspondingly, SPaths) may be exhausted or fragmented in such a way that the embedding of $\bar{e}_i$ becomes infeasible. Note that VLinks that come after $\bar{e}_i$ in o do not have any impact on the embedding of $\bar{e}_i$ even though $\bar{e}_i$ and the VLinks after $\bar{e}_i$ have common SLinks in their candidate SPath sets. Hence, effective commonality of $\bar{e}_i$ depends on o and is defined as follows:

$$w(\bar{e}_i)^o = \sum_{\bar{e}_j \text{ precedes } \bar{e}_i \text{ in } o | \bar{e}_i \in \bar{E},\, \bar{e}_j \in \bar{E},\, \bar{e}_i \neq \bar{e}_j} w(\bar{e}_i, \bar{e}_j) \tag{5.9}$$

where, $w(\bar{e}_i, \bar{e}_j)$ is the commonality between two VLinks $\bar{e}_i$ and $\bar{e}_j$ irrespective of any order and is defined as follows:

$$w(\bar{e}_i, \bar{e}_j) = |\{(p_{\bar{e}_i}, p_{\bar{e}_j}) : (p_{\bar{e}_i}, p_{\bar{e}_j}) \in \mathcal{P}^k_{\bar{e}_i} \times \mathcal{P}^k_{\bar{e}_j} \wedge p_{\bar{e}_i} \cap p_{\bar{e}_j} \neq \phi\}| \tag{5.10}$$

A high value of $w(\bar{e}_i, \bar{e}_j)$ indicates more SPath pairs from the candidate SPath sets of $\bar{e}_i$ and $\bar{e}_j$ have common SLinks. Using (5.9), we define the commonality index of o as follows:

$$w^o = \max_{\bar{e}_i \in \bar{E}} w(\bar{e}_i)^o \tag{5.11}$$

To increase the chances of finding a feasible embedding for all the VLinks, Alg. 12 finds an order O that minimizes w^o for all possible orders of $\bar{E}$. To do so, Alg. 12 first constructs an auxiliary graph $Z = (N, A)$ based on a VN $\bar{G}$ and EON G (lines 2). The auxiliary graph $Z = (N, A)$ is a weighted undirected graph where N is the set of nodes and A is the set of edges. There is a one-to-one correspondence between a node $n_{\bar{e}_i} \in N$ and a VLink $\bar{e}_i \in \bar{E}$. Therefore, an order of the nodes in N corresponds to an order of the VLinks in $\bar{E}$. We include a weighted edge $(n_{\bar{e}_i}, n_{\bar{e}_j}) \in A$ between two distinct nodes $n_{\bar{e}_i} \in N$ and $n_{\bar{e}_j} \in N$ with weight $w(\bar{e}_i, \bar{e}_j)$. Using (5.10), we define the weight of a node $n_{\bar{e}_i} \in N$ as follows:

$$w(n_{\bar{e}_i}) = \sum_{n_{\bar{e}_j} | (n_{\bar{e}_i}, n_{\bar{e}_j}) \in A} w(\bar{e}_i, \bar{e}_j) \tag{5.12}$$

To compute a node order (or a corresponding VLink order) that minimizes w^o, Alg. 12 iteratively computes $w(n_{\bar{e}_i})$ using (5.12) for all the nodes in N. Then Alg. 12 chooses the node $n^{min}_{\bar{e}_i}$ with the minimum weight $w(n_{\bar{e}_i})$ and inserts the corresponding VLink to the last empty spot in the VLink order O (line 15). Afterwards, the algorithm updates Z by removing $n^{min}_{\bar{e}_i}$ and all of its incident edges. The updated auxiliary graph Z allows us to use (5.12) to compute $w(n_{\bar{e}_i})$ that corresponds to $w(\bar{e}_i)^O$ as updated Z now only have all the nodes (and adjacent edges) that precede $n^{min}_{\bar{e}_i}$ in the node order, or equivalently $\bar{e}^{min}_i$ in O. Alg. 12 repeats this process until all the VLinks are added to O.

Theorem 1. *Alg. 12 returns a VLink ordering with the minimum commonality index.*

Proof. Suppose VLink ordering o which is generated by Alg. 12 does not have the minimum commonality index, therefore, there exists a VLink ordering o^* for which $w^o > w^{o^*}$. Let $\bar{e}^o_1, \bar{e}^o_2, ..., \bar{e}^o_{|\bar{E}|}$ and $\bar{e}^{o^*}_1, \bar{e}^{o^*}_2, ..., \bar{e}^{o^*}_{|\bar{E}|}$ denote the VLink ordering o and o^* respectively. Since $w^o \neq w^{o^*}$ there exists at least one index i for which $\bar{e}^o_j$ and $\bar{e}^{o^*}_j$ are not corresponding to the same VLink. Let i_{max} be the maximum index with this condition. Since both VLink ordering o and o^* contain all the VLinks, there should be an index j such that $\bar{e}^{o^*}_j$ corresponds to the same VLink as $\bar{e}^o_{i_{max}}$. We create a new VLink ordering o^*_1 by moving the $\bar{e}^{o^*}_j$ to the i_{max}th index in the o^* VLink ordering (Fig. 5.2).

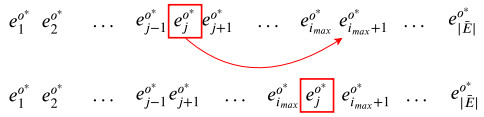


Figure 5.2: Creating VLink ordering o^*_1 from o^*

Lemma 1. *The commonality index of VLink ordering o^*_1 is less than or equal to the commonality index of the VLink ordering o^* ($w^{o^*_1} \leq w^{o^*}$).*

Proof. We divide the proof of the lemma in two parts. At first, we prove *i)* the effective commonality of all the VLinks (except $\bar{e}_j^{o^*}$) in VLink ordering o_1^* is less than or equal to the effective commonality of the same VLink in VLink ordering o^*, i.e. $\forall \bar{e}_i \in \bar{E} - \{\bar{e}_j^{o^*}\} : w(\bar{e}_i)^{o_1^*} \leq w(\bar{e}_i)^{o^*}$. Then we prove *ii)* $w(\bar{e}_j^{o^*})^{o_1^*} \leq w(\bar{e}_{i_{max}}^{o^*})^{o^*}$, hence, for each VLink in VLink ordering o_1^* there exists at least one VLink in VLink ordering o^* such that its effective commonality is greater or equal, i.e. $\forall \bar{e}_i \in \bar{E}, \exists \bar{e}_j : w(\bar{e}_i)^{o_1^*} \leq w(\bar{e}_j)^{o^*}$. From (5.11), the commonality index of a VLink ordering is equal to the maximum effective commonality of all VLinks in that ordering, therefore, $w^{o_1^*} \leq w^{o^*}$.

To prove *i)*, we divide VLinks into 3 groups (Fig. 5.3):

1. $A = \{\bar{e}_i^{o^*} | 1 \leq i < j\}$ which includes all the VLinks that come before $\bar{e}_j^{o^*}$ in both o^* and o_1^*.
2. $B = \{\bar{e}_i^{o^*} | j < i \leq i_{max}\}$ which contains all the VLinks that come after $\bar{e}_j^{o^*}$ in o^*, but before $\bar{e}_j^{o^*}$ in o_1^*.
3. $C = \{\bar{e}_i^{o^*} | i_{max} < i \leq |\bar{E}|\}$ which includes the VLinks that come after $\bar{e}_j^{o^*}$ in both o^* and o_1^*.

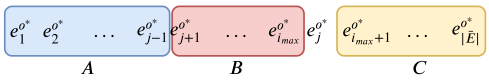


Figure 5.3: Dividing the VLinks into 3 groups to check their commonality indexes

From (5.9), we know that the effective commonality of each VLink only depends on the preceding VLinks. Hence, considering that the set of preceding VLinks is the same $\forall \bar{e}_i \in A \cup C$ in ordering o^* and o_1^*, their effective commonalities are the same, i.e. $\forall \bar{e}_i \in A \cup C : w(\bar{e}_i^{o^*})^{o_1^*} = w(\bar{e}_i^{o^*})^{o^*}$. In addition, the set of preceding VLinks of $\bar{e}_i \in B$ in o_1^* is the same as their set of preceding VLinks in o^* minus $\bar{e}_j^{o^*}$. Therefore, $\forall \bar{e}_i \in B : w(\bar{e}_i)^{o_1^*} = w(\bar{e}_i)^{o^*} - w(\bar{e}_i, \bar{e}_j^{o^*}) \leq w(\bar{e}_i)^{o^*}$.

For proving *ii)*, we need to show that $w(\bar{e}_j^{o^*})^{o_1^*} \leq w(\bar{e}_{i_{max}}^{o^*})^{o^*}$. Recall that Alg. 12, at each iteration, finds the VLink with minimum weight and puts it at the first empty place at the end of ordering (Line 4–7). Since the ordering o, which is generated by Alg. 12, is identical to ordering o^* from index $i_{max}+1$ to $\bar{E}$, it means that VLink $\bar{e}_j^o = \bar{e}_{i_{max}}^{o^*}$ has the minimum weight, therefore $w(n_{\bar{e}_j^o}) \leq w(n_{\bar{e}_{i_{max}}^{o^*}})$ at $(n - i_{max} + 2)$-th step (to find the VLink that should be at index i_{max}). Since (5.12) computes the effective commonality of the all the remaining VLinks assuming they will be at the last empty spots $w(n_{\bar{e}_j^o}) = w(\bar{e}_j^{o^*})^{o_1^*}$ and $w(n_{\bar{e}_{i_{max}}^{o^*}}) = w(\bar{e}_{i_{max}}^{o^*})^{o^*}$. So $w(\bar{e}_j^{o^*})^{o_1^*} \leq w(\bar{e}_{i_{max}}^{o^*})^{o^*}$.

By showing these two parts, we prove that $w^{o_1^*} \leq w^{o^*}$. □

We start from VLink ordering $o_0^* = o^*$ and at ith step we generate o_i^* from o_{i-1}^* using the same approach to generate o_1^* from o^*. Since the maximum index for which o_1^* and o are different decreases after each step, finally, after $s \leq |\bar{E}|$ steps, $o = o_s^*$. By 1 we know that the commonality index does not increase after each step, i.e., $w^{o_i^*} \leq w^{o_{i-1}^*}$. Therefore $w^o = w^{o_s^*} \leq w^{o^*}$, which contradicts the initial assumption. This means that Alg. 12 returns an ordering with the minimum commonality index. □

5.4.3 Compute Embedding of a Single VLink

The embedding of a VLink computed by Alg. 13 consists of a multi-set of SPaths where each SPath in the multi-set has an associated transmission configuration and spectrum slot allocation. Recall from Fig. 5.1 that bandwidth requirement for dedicated protection against single SLink failure for a VLink depends on BSR and the number of disjoint SPaths used in the multi-set of SPaths of a solution. To exploit disjointness of the SPaths in the candidate SPath set $\mathbb{P}_{\bar{e}}^k$ of a VLink $\bar{e}$, Alg. 13 first computes disjoint SPath groups from the SPaths in $\mathbb{P}_{\bar{e}}^k$. We define a disjoint SPath group $H_{\bar{e}}$ from $\mathcal{P}_{\bar{e}}^k$ as follows:

$$\begin{aligned} H_{\bar{e}} = \{\delta\mathcal{P}_{\bar{e}}^k | \delta\mathcal{P}_{\bar{e}}^k \subseteq \mathcal{P}_{\bar{e}}^k \text{and } |\delta\mathcal{P}_{\bar{e}}^k| > 1 \text{ and} \\ \text{the SPaths in } \delta\mathcal{P}_{\bar{e}}^k \text{ are link disjoint}\} \end{aligned} \tag{5.13}$$

Note that two SPaths belonging to two different disjoint SPath groups $H_{\bar{e}}^i$ and $H_{\bar{e}}^j$ may share an SLink and an SPath can appear in multiple disjoint SPath groups. The set of all disjoint SPath groups for a VLink $\bar{e}$ is denoted by $\mathcal{H}_{\bar{e}}$. For instance, in Fig. 5.1(a), $\mathcal{H}_{\bar{pq}} = \{\{AB-BC, AD-DC\}, \{AB-BC, AE-EC\}, \{AD-DC, AE-EC\}, \{AB-BC, AD-DC, AE-EC\}\}$, and in Fig. 5.1(e), $\mathcal{H}_{\bar{pq}} = \{\{AD-DC, AE-EC\}, \{AD-DB-BC, AE-EC\}\}$.

Alg. 13 needs to enumerate all non-empty subsets of $\mathcal{H}_{\bar{e}}$ to find the optimal solution. Such enumeration is not scalable as the number of subsets of $\mathcal{H}_{\bar{e}}$ grows exponentially with the size of $\mathcal{H}_{\bar{e}}$. Hence, Alg. 13 applies a heuristic to obtain a smaller set of disjoint SPath groups $\mathbb{H}_{\bar{e}} \subseteq \mathcal{H}_{\bar{e}}$ that includes the most probable SPaths to be used as the splits of $\bar{e}$. The heuristic is motivated by the fact that longer SPaths allow only lower order modulation formats with a higher FEC that may require a large number of spectrum slots. In addition, longer SPaths often consist of more intermediate hops, thus increasing the total spectrum usage as per (5.8). To exclude such longer SPaths, our heuristic should construct $\mathbb{H}_{\bar{e}}$ with those disjoint SPath groups whose average total distances are small. However, doing so may bias the algorithm to include disjoint SPath groups with smaller number of SPaths for keeping the average distance low, which can reduce the benefits of splitting over multiple

disjoint paths. Alg. 13 circumvents this issue by considering disjoint SPath groups of all sizes. From each set of disjoint SPath groups with size i, Alg. 13 selects the first σ disjoint SPath groups ranked by the increasing average distance of the group (Line 6–9). The value of σ is an input to Alg. 13 that can be used to keep the size of $\mathbb{H}_{\bar{e}}$ small.

After computing $\mathbb{H}_{\bar{e}}$, Alg. 13 enumerates all non-empty subsets of $\mathbb{H}_{\bar{e}}$ to assign data-rates to each disjoint SPath group in the subset such that each group provides dedicated protection as per BSR requirement for its data-rate. For each subset $\delta\mathbb{H}_{\bar{e}} \subseteq \mathbb{H}_{\bar{e}}$, Alg. 13 selects $|\delta\mathbb{H}_{\bar{e}}|$ data-rates such that the sum of these data-rates equals to demand $\bar{\beta}_{\bar{e}}$ (Line 10). These combinations of data-rates is represented by a multi-set $\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}} = (\mathcal{D}, m_1)$, where $\mathcal{D}$ is the set of all data-rates, and $m_1 : \mathcal{D} \rightarrow N$ is the number of times a data-rate in $\mathcal{D}$ appears in $\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}}$. $\mathcal{M}(\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}})$ represents all possible multi-sets of $\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}}$.

Since none of $\delta\mathbb{H}_{\bar{e}}$ and $\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}}$ are ordered sets, Alg. 13 needs to enumerate all permutations of data-rates from $\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}}$ to assign data-rates in $\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}}$ to $|\delta\mathbb{H}_{\bar{e}}|$ SPath groups. To do so, Alg. 13 generates all permutations of data-rates for each multi-set $\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}} \in \mathcal{M}(\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}})$, denoted by $\zeta(\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}})$ (Line 12). For each of these permutations of data-rates $\mathrm{D}_{\delta\mathbb{H}_{\bar{e}}} \in \zeta(\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}})$, Alg. 13 assigns a data-rate $d_{H_{\bar{e}}}$ from $\mathrm{D}_{\delta\mathbb{H}_{\bar{e}}}$ to a disjoint SPath group $H_{\bar{e}} \in \delta\mathbb{H}_{\bar{e}}$ in the same order (Line 15). For each disjoint SPath group $H_{\bar{e}} \in \delta\mathbb{H}_{\bar{e}}$ with its assigned data-rate $d_{H_{\bar{e}}}$, the algorithm computes the data-rate $d_{p_{H_{\bar{e}}}}$ of an SPath $p \in H_{\bar{e}}$ using (5.14) that takes into account the number of disjoint SPaths in $H_{\bar{e}}$ and $BSR_{\bar{e}}$ of the VLink (Line 16). $d_{p_{H_{\bar{e}}}}$ is computed in a way such that the SPaths in $H_{\bar{e}}$ protect $BSR_{\bar{e}}$ fraction of $d_{H_{\bar{e}}}$ under any single SLink failure. However, when $BSR_{\bar{e}}$ is low, protection requirement is not enough to provide the assigned data-rate $d_{H_{\bar{e}}}$. In this case, data-rate $d_{p_{H_{\bar{e}}}}$ for an SPath is computed by equally dividing $d_{H_{\bar{e}}}$ along each disjoint SPath in $H_{\bar{e}}$. Note that each SPath in $H_{\bar{e}}$ will have an equal data-rate $d_{p_{H_{\bar{e}}}}$.

$$d_{p_{H_{\bar{e}}}} = max\left(\frac{d_{H_{\bar{e}}} \times BSR_{\bar{e}}}{100 \times (|H_{\bar{e}}| - 1)}, \frac{d_{H_{\bar{e}}}}{|H_{\bar{e}}|}\right) \tag{5.14}$$

Since an SPath p can appear in multiple disjoint SPath groups in $\delta\mathbb{H}_{\bar{e}}$, a consolidation step is introduced to compute the total data-rate assigned to p (denoted by $d_{p_{\delta\mathbb{H}_{\bar{e}}}}$) using (5.15) (Line 17). Since the possible data-rates in the reach table are discrete values, the ceiling function returns the nearest rounded up data-rate after the summation in (5.15).

$$d_{p_{\delta\mathbb{H}_{\bar{e}}}} = \left\lceil \sum_{\forall H_{\bar{e}} \in \delta\mathbb{H}_{\bar{e}}} d_{p_{H_{\bar{e}}}} \right\rceil \tag{5.15}$$

After the consolidation step, we get a set of distinct (not necessarily disjoint) SPaths $P_{\delta\mathbb{H}_{\bar{e}}}$ and their assigned data-rates for a particular $\delta\mathbb{H}_{\bar{e}}$ and $\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}}$. However, an SPath

$p_{\delta\mathbb{H}_{\bar{e}}} \in P_{\delta\mathbb{H}_{\bar{e}}}$ may split its assigned data-rate $d_{p_{\delta\mathbb{H}_{\bar{e}}}}$ into smaller data-rates either to ensure spectrum contiguity or to minimize the number of slots. To enumerate these possibilities, Alg. 13 generates the set of all possible multi-sets $\mathcal{M}(\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}})$ out of the set $P_{\delta\mathbb{H}_{\bar{e}}}$ (Line 18). A multi-set $\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}} \in \mathcal{M}(\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}})$ is defined as $\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}} = (P_{\delta\mathbb{H}_{\bar{e}}}, m_2)$, where, $m_2 : P_{\delta\mathbb{H}_{\bar{e}}} \rightarrow N$ is the number of times an SPath in $P_{\delta\mathbb{H}_{\bar{e}}}$ appears in $\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}$. For each multi-set $\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}$, Alg. 13 assigns data-rates to each instance of SPath in $\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}$. This is trivial for an SPath that appears once in $\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}$ ($m_2(p) = 1$). For an SPath with $m_2(p) > 1$, Alg. 13 distributes the assigned data-rate $d_{p_{\delta\mathbb{H}_{\bar{e}}}}$ into $m_2(p)$ splits by generating permutations of multi-sets of data-rates of size $m_2(p)$ (Line 21–23). To get the set of all data-rate permutations $\mathcal{M}(\mathrm{D}_{\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}})$ for $\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}$, Alg. 13 considers all possible ways to combine the data-rate permutations of each distinct SPath in $P_{\delta\mathbb{H}_{\bar{e}}}$ (Line 24).

Once we have an SPath multi-set $\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}$ and its data-rate permutation $\mathrm{D}_{\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}}$, Alg. 13 invokes MDP procedure to find feasible transmission configuration and spectrum slot allocation with the least slot requirement (Line 26). MDP procedure is adapted from [147] that first selects a transmission configuration to achieve a data-rate in $\mathrm{D}_{\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}}$ along an SPath in $\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}$ and allocates the slots required by the transmission configuration using First-fit [38]. Among all the feasible solutions returned by MDP procedure, Alg. 13 selects the one that minimizes (5.8).

Running Time Analysis

Alg. 13 explores all subsets of $\mathbb{H}_{\bar{e}}$ yielding $2^{|\mathbb{H}_{\bar{e}}|}$ possibilities. For each subset $\delta\mathbb{H}_{\bar{e}} \subseteq \mathbb{H}_{\bar{e}}$, Alg. 13 explores $\binom{|\mathcal{D}|+|\delta\mathbb{H}_{\bar{e}}|-1}{|\delta\mathbb{H}_{\bar{e}}|}$ data-rate multi-sets. The number of permutations of a multiset $\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}}$ of cardinality $|\delta\mathbb{H}_{\bar{e}}|$ is given by $\frac{|\delta\mathbb{H}_{\bar{e}}|!}{\Pi_{d_j \in \mathcal{D}} m_1(d_j)!}$ [29]. This results in $\frac{(|\mathcal{D}|+|\delta\mathbb{H}_{\bar{e}}|-1)!}{(|\mathcal{D}|-1)! \times \Pi_{d_j \in \mathcal{D}} m_1(d_j)!}$ enumerations. Then Alg. 13 enumerates $\mathcal{M}(\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}})$ multi-sets of SPaths based on an SPath set $P_{\delta\mathbb{H}_{\bar{e}}}$, and this enumeration has an upper bound $\binom{q-1}{\frac{q}{2}} = \frac{\Pi_{j=\lceil \frac{q}{2} \rceil + 1}^{q-1} j!}{\lceil \frac{q}{2} \rceil !}$. Assuming that for each SPath $p \in P_{\delta\mathbb{H}_{\bar{e}}}$, Alg. 13 can have a data-rate multi-set $\mathbb{D}_p = (\mathcal{D}, m_3)$, where $m_3 : \mathcal{D} \rightarrow N$ is the frequency of a data-rate in $\mathbb{D}_p$, the number of data-rate permutations in $\mathcal{M}(\mathrm{D}_{\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}})$ is $\Pi_{i=1}^{|P_{\delta\mathbb{H}_{\bar{e}}}|} \frac{(|\mathcal{D}|+m_2(p_i)-1)!}{(|\mathcal{D}|-1)! \times \Pi_{d_j \in \mathcal{D}} m_3(d_j)!}$. Since MDP's time complexity is $\frac{q!}{\Pi_{p_j \in \mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}} m_2(p_j)!}$ as per [147], the running time of Alg. 13 becomes $2^{|\mathbb{H}_{\bar{e}}|} \times \frac{(|\mathcal{D}|+|\delta\mathbb{H}_{\bar{e}}|-1)!}{(|\mathcal{D}|-1)! \times \Pi_{d_j \in \mathcal{D}} m_1(d_j)!} \times \frac{\Pi_{j=\lceil \frac{q}{2} \rceil + 1}^{q-1} j!}{\lceil \frac{q}{2} \rceil !} \times \Pi_{i=1}^{|P_{\delta\mathbb{H}_{\bar{e}}}|} \frac{(|\mathcal{D}|+m_2(p_i)-1)!}{(|\mathcal{D}|-1)! \times \Pi_{d_j \in \mathcal{D}} m_3(d_j)!} \times \frac{q!}{\Pi_{p_j \in \mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}} m_2(p_j)!}$. As Alg. 13 keeps the size of $\mathbb{H}_{\bar{e}}$ small, the running time is dominated by the latter part. However, typical values of $|\mathcal{D}|$ and q are small and we apply several pruning techniques to improve the running time.

Algorithm 13: Find the embedding of a single VLink

function *FindEmbedding*$(G, \bar{e}, \mathcal{P}_{\bar{e}}^{k}, \mathcal{T}_{\bar{e}}, \sigma)$
  while *A new* $H_{\bar{e}}$ *exists* **do**
    Compute a new disjoint path group $H_{\bar{e}}$ using (5.13)
    $\mathcal{H}_{\bar{e}} \leftarrow \mathcal{H}_{\bar{e}} \cup H_{\bar{e}}$
  for $i = 2$ *to* $\max_{\forall H_{\bar{e}} \in \mathcal{H}_{\bar{e}}} |H_{\bar{e}}|$ **do**
    $Z_{\bar{e}}^{i} \leftarrow \{H_{\bar{e}} | H_{\bar{e}} \in \mathcal{H}_{\bar{e}} \wedge |H_{\bar{e}}| = i\}$
    $\mathcal{Z}_{\bar{e}}^{i} \leftarrow$ First σ $H_{\bar{e}} \in Z_{\bar{e}}^{i}$ with smallest avg. dist.
    $\mathbb{H}_{\bar{e}} \leftarrow \mathbb{H}_{\bar{e}} \cup \mathcal{Z}_{\bar{e}}^{i}$
  foreach $\delta\mathbb{H}_{\bar{e}} \subseteq \mathbb{H}_{\bar{e}}$ **do**
    $\mathcal{M}(\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}}) \leftarrow$ All-Multi-Set$(\mathcal{D}, |\delta\mathbb{H}_{\bar{e}}|)$ s.t. $\sum_{d \in \mathbb{D}_{\delta\mathbb{H}_{\bar{e}}}} d = \bar{\beta}_{\bar{e}}$
    foreach $\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}} \in \mathcal{M}(\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}})$ **do**
      $\zeta(\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}}) \leftarrow$ All-Permutation$(\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}})$
      foreach $\mathrm{D}_{\delta\mathbb{H}_{\bar{e}}} \in \zeta(\mathbb{D}_{\delta\mathbb{H}_{\bar{e}}})$ **do**
        foreach $H_{\bar{e}} \in \delta\mathbb{H}_{\bar{e}}$ **do**
          $d_{H_{\bar{e}}} \leftarrow \mathrm{D}_{\delta\mathbb{H}_{\bar{e}}}$[index of $H_{\bar{e}} \in \delta\mathbb{H}_{\bar{e}}$]
          Compute $d_{p_{H_{\bar{e}}}}$ using (5.14)
        Compute $d_{p_{\delta\mathbb{H}_{\bar{e}}}}$ using (5.15)
        $\mathcal{M}(\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}) \leftarrow$ All-Multi-Set$(P_{\delta\mathbb{H}_{\bar{e}}}, q)$
        foreach $\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}} \in \mathcal{M}(\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}})$ **do**
          foreach $p \in \mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}$ **do**
            $\mathcal{M}(\mathbb{D}_p) \leftarrow$ All-Multi-Set$(\mathcal{D}, m_2(p))$ s.t. $\sum_{d \in \mathbb{D}_p} d = d_{p_{\delta\mathbb{H}_{\bar{e}}}}$
            foreach $\mathbb{D}_p \in \mathcal{M}(\mathbb{D}_p)$ **do**
              $\zeta(\mathbb{D}_p) \leftarrow$ All-Permutation$(\mathbb{D}_p)$
          $\mathcal{M}(\mathrm{D}_{\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}}) \leftarrow \zeta(\mathbb{D}_{p_1}) \times \zeta(\mathbb{D}_{p_2}) ... \times \zeta(\mathbb{D}_{p_{|P_{\delta\mathbb{H}_{\bar{e}}}|}})$
          foreach $\mathrm{D}_{\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}} \in \mathcal{M}(\mathrm{D}_{\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}})$ **do**
            $< n, \mathbb{P}, \mathbb{T}, \mathbb{S} > \leftarrow$ MDP$(\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}, \mathrm{D}_{\mathbb{P}_{\delta\mathbb{H}_{\bar{e}}}})$
  Find $\mathbb{P}^{opt}$, $\mathbb{T}^{opt}$ and $\mathbb{S}^{opt}$ that minimizes (5.8)
  $\mathcal{I}_{\bar{e}} \leftarrow < \mathbb{P}^{opt}, \mathbb{T}^{opt}, \mathbb{S}^{opt} >$
  return $\mathcal{I}_{\bar{e}}$

5.5 Evaluation

We evaluate the proposed solutions through extensive simulations. § 5.5.1 describes the simulation setup in detail and § 5.5.2 defines the performance metrics. Then we present

our evaluation results focusing on the following three aspects. First, we perform micro-benchmarks of our solutions and compare with the demand splitting model presented in [66]), a recent work on survivable VNE over flexible grid optical networks. For the micro-benchmark scenario, we consider each VN embedding request in isolation and assume that the VN can always be embedded on the EON. Under these assumptions, we measure the resource efficiency of our proposed solutions. Second, we analyze the performance of the heuristic algorithm in both small and large scale settings considering each VN embedding request in isolation. Finally, we perform steady state analysis of our heuristic solution. The steady state analysis considers VN arrivals and departures over a period of time and also considers the possibility of failing to embed a VN request on the EON. The steady state analysis provides valuable insights into the number of accepted VNs and the EON link utilization in a longer time frame.

5.5.1 Simulation Setup

Testbed

We implement the ILP formulation from § 5.3 using IBM ILOG CPLEX C++ libraries. We compare the ILP's solution with a C++ implementation of the heuristic presented in § 5.4. Simulations are run on a machine with 8×10-core Intel Xeon E7-8870 2.40GHz processors and 1TB RAM. We developed a discrete event simulator that simulates the arrival and departure of VNs for the steady state scenario.

SN characteristic

We use Nobel Germany (17 nodes and 26 links) and Germany50 (50 nodes and 88 links) networks from SNDlib repository [7] as the EONs for small and large scale simulations, respectively. We pre-compute $k = 25$ and $k = 20$ shortest paths between all pairs of SNodes as inputs to our simulation for Nobel Germany and Germany50 SN, respectively. Each SLink in the EON has spectrum bandwidth of 600GHz and 4THz for small and large scale simulations, respectively [4]. We also vary the Link-to-Node Ratio (LNR) of Nobel Germany SN by adding or removing some links in the original topology (original LNR is 1.53). This is done to show the impact of different levels of SN densities on the performance metrics. For steady state analysis, we use Nobel Germany topology as the EON and set 4THz spectrum bandwidth on each SLink.

VN generation

We synthetically generate VNs with different properties. For the small-scale scenario, we restrict VN sizes to 4 VNodes and 5 VLinks to limit the ILP's complexity. For the large-scale scenario, we generate VNs with 20 VNodes and 30 VLinks. We set VLink demands to multiples of 100Gbps, ranging between 100Gbps and 1Tbps, following the granularity of commercial transponders. Node mapping of VNodes to SNodes is randomly chosen. In our simulations, we vary BSR from 0% to 100% with 20%-point increments. Here, $BSR = 0$ means no protection and $BSR = 100\%$ provides full dedicated protection. We generate 5 and 20 different VNs with similar total bandwidth demands for small- and large-scale simulations, respectively, and take the mean of the performance metrics over those VNs.

For the steady state analysis, we simulate VN arrival and departure scenarios with different arrival rates. The VN arrival rate of each scenario follows a Poisson distribution. We vary the mean of the Poisson distribution between $4 - 12$ VNs per 100 time units. VN life time in all the scenarios is exponentially distributed with a mean of 100 time units. We generate each arrived VN with random connectivity between VNodes. We keep the number of VNodes of each arrived VN at 4, and chose the LNR of the VN randomly between 1 and 3.5. VLink demands are chosen in the same way as in the micro-benchmark scenario.

To demonstrate the impact of different BSR requirements on the blocking ratio and at the same time to simulate the behavior of different VN characteristics and requirements, we generate BSR of each VN using a Binomial distribution. For each arrival rate, we generate 5 simulation scenarios, each with a different Binomial distribution of BSRs. For these 5 scenarios, we set the mean of the Binomial distribution to 0, 20, 40, 60, and 80%-points, respectively. We run the simulation for 10000 simulated time units and exclude measurements from the first 1000 time units to capture the steady state performance. For each problem instance with a fixed arrival rate and mean BSR, we generate 5 random simulation scenarios and report the mean performance metrics to gain statistical confidence. These parameters have been chosen in accordance with those used in the network virtualization literature (*e.g.*, [44, 50, 142]).

5.5.2 Compared Variants

In our evaluation, we quantify the impact of the two key flexibilities introduced by EONs, namely, adaptability in transponder configuration (*i.e.*, variable FEC and modulation) and in spectrum allocation. In doing so, we compare the variants listed in Table 5.2. *Fix-RT* considers fixed-grid spectrum allocation with only one tuple for modulation format,

Table 5.2: Compared Variants

Variant	**Transponder Flexibility**	**Flexible Spectrum Allocation**
Fix-RT	No	No
Fix-AT	Yes	No
Flex-AT	Yes	Yes

FEC overhead, and reach for each fixed-grid data rate in $\mathcal{R}$ and serves as the baseline for our evaluation. In contrast, the other two variants, *Fix-AT* and *Flex-AT*, exploit a transponder's capability to choose from a variety of modulation formats and/or FEC overheads for each data rate in $\mathcal{R}$, leading to different reaches. These two variants differ in terms of spectrum slot allocation granularity (*i.e.*, 50GHz and 12.5GHz for fixed-grid and flex-grid, respectively) and the total number of possible data rates (*i.e.*, 100G, 200G, 400G for fixed-grid and 100G, 150G, 200G, 250G, 300G, 400G, 500G, 600G, and 800G for flex-grid). Spectrum occupation and data rates of each variant are chosen based on current industry standards [4]. We set $q = 8$ to keep the number of used transponders reasonable.

5.5.3 Performance Metrics

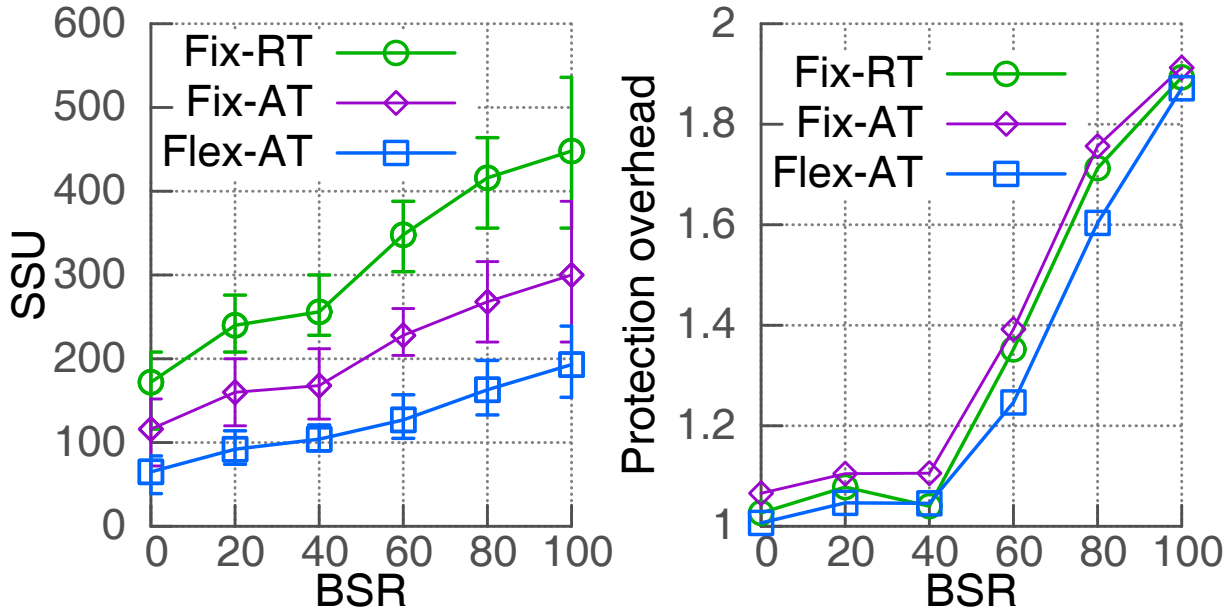
Spectrum slot Usage (SSU) The total number of spectrum slots required to embed a VN. This metric is computed using only the first term of (5.8).

Protection overhead Ratio between total bandwidth allocated of a VLink (on all different splits) and the actual bandwidth demand of the VLink. It is a measure of the required extra bandwidth for providing protection.

Max. no. of disjoint paths Average of the maximum number of disjoint SPaths used to embed a VLink.

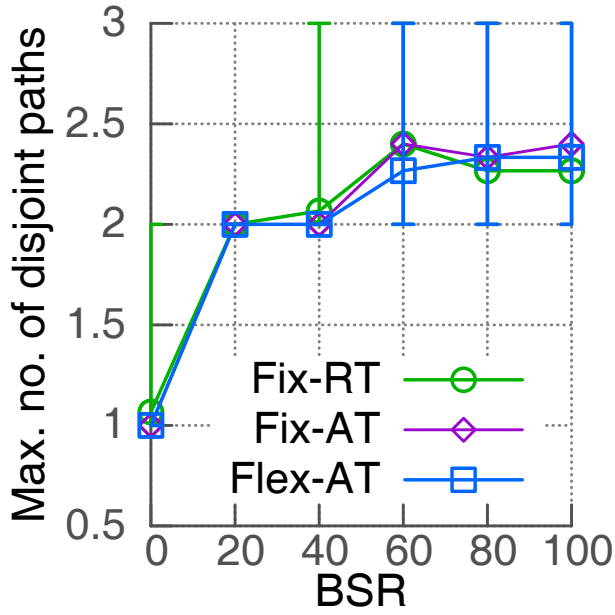
Max. no. of splits Average of the maximum number of splits used to embed a VLink.

Cost ratio The ratio of costs obtained by two different approaches for solving the same problem instance, where cost is computed using (5.8).

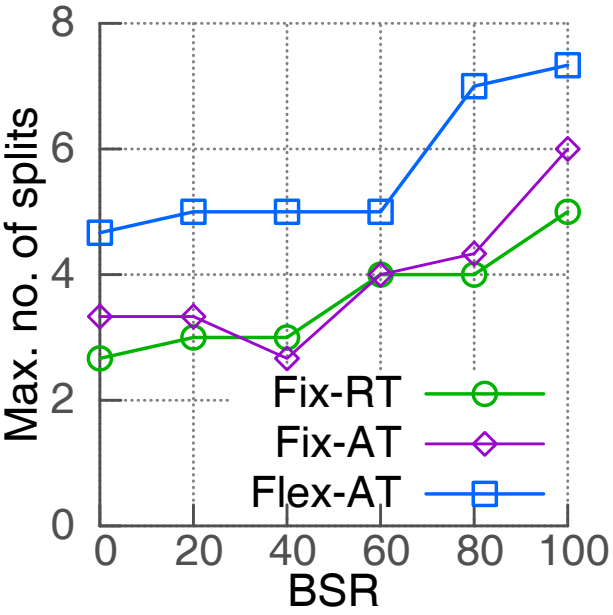


(a) Impact on SSU

(b) Impact on protection overhead



(c) Impact on number of disjoint paths



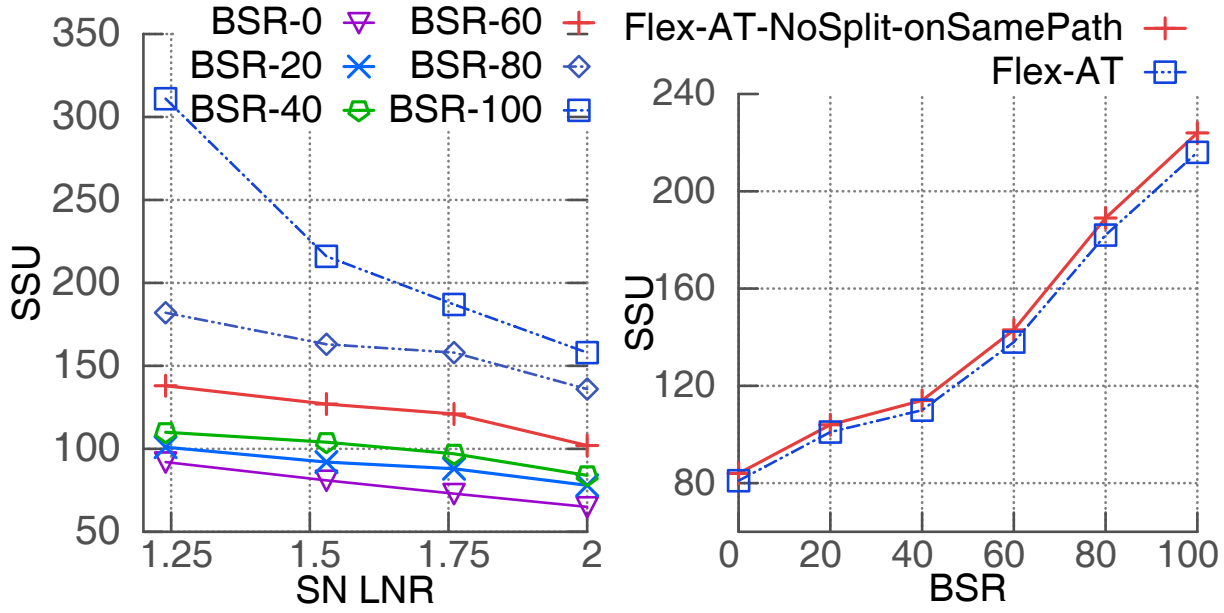
(d) Impact on number of splits

Figure 5.4: Impact of varying BSR on performance metrics in Nobel Germany EON

Execution time The time required for an algorithm to find a VN embedding.

Blocking ratio The fraction of VN requests that could not be embedded on the EON over all the VN requests in a simulation.

Link utilization The ratio between the number of slots used in an SLink and the total number of slots in the SLink at any instance of time.



(a) Impact of SN density

(b) Split vs noSplit on same path

Figure 5.5: Analysis of our reliability model using the Nobel Germany EON

5.5.4 Micro-benchmark results

Small-scale results with ILP formulation

Fig. 5.4 presents the impact of varying BSR on all performance metrics, and for all the compared variants, in Nobel Germany SN. Specifically, Fig. 5.4(a) shows that SSU is the highest for the baseline variant, *Fix-RT*, since it does not allow any flexibility in tuning transmission parameters or spectrum allocation. In contrast, *Fix-AT* uses 30% less

spectrum resources on average compared to *Fix-RT* by only using tunable transponders with coarse-grained spectrum allocation. By combining flexible spectrum allocation with transponder tunability, *Flex-AT* saves on average 57% and 27% spectrum compared to *Fix-RT* and *Fix-AT*, respectively. Lastly, *Fix-RT* was infeasible for some problem instances at BSR > 80 due to its inflexibility in transmission configuration tuning and spectrum allocation.

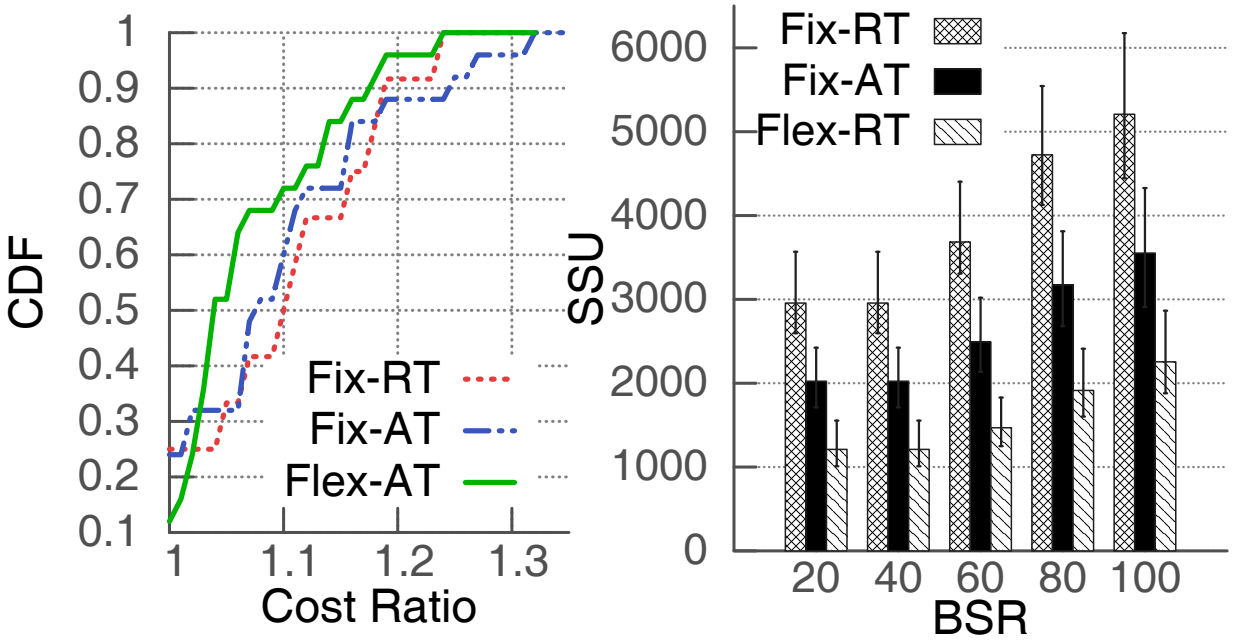
Fig. 5.4(a) also shows that SSU increases for increasing BSR in all considered cases. However, SSU rises very slowly (remaining within 10% of the originally requested bandwidth) when BSR $\leq 40\%$, while it steeply increases for BSR $> 40\%$. 10% additional bandwidth is due to the lack of fine-grained data rates that results in a small amount of over-provisioning (*e.g.*, 350Gbps is needed for BSR=40% and demand=800Gbps, as 320Gbps is not a valid data rate). This can be explained by observing how the protection overhead (see plot in Fig. 5.4(b)) behaves for BSR up to 40%. This behavior of the protection overhead for BSR $\leq 40\%$ confirms the intuition, as shown in Fig. 5.1, that low BSR can be guaranteed with low or no additional bandwidth if at least two disjoint SPaths can be used to map the splits of a VLink. In contrast, for BSR $> 40\%$, more than two disjoint SPaths are needed for each VLink to reduce protection overhead. However, in our SN (Nobel Germany), either more than two disjoint SPaths do not exist for some pairs of SNodes, or the third and higher order disjoint SPaths become too long due to the sparse connectivity of the SN. Since long SPaths can support only low modulation formats, and thus require high number of spectrum slots, they are very unlikely to be selected in the optimal solution. This can be verified in Fig. 5.4(c), where it can be seen that the max. no. of disjoint SPaths used to embed a VLink is slightly larger than two for BSR $> 40\%$, thus increasing protection overhead and, eventually, SSU.

Fig. 5.4(d) shows that the max. no. of splits of a VLink is always larger than the max. no. of disjoint SPaths of a VLink, implying that there are multiple splits on the same SPath. Note also that Fig. 5.4(d) shows that *Flex-AT* uses the highest number of splits to minimize the SSU as shown in Fig. 5.4(a). This is due to *Flex-AT*'s smaller granularity in spectrum allocation that allows to use shorter SPaths even if the spectrum resources on these SPaths are fragmented.

To demonstrate how SN connectivity impacts spectral efficiency, Fig. 5.5(a) shows SSU for *Flex-AT* with different BSR requirements by varying LNR of Nobel Germany SN. Fig. 5.5 shows that SSUs decrease with increasing SN LNR even for the no protection case (0% BSR). This stems from the fact that an SN with a larger LNR reduces the lengths and the number of intermediate hops of the SPaths between the same pair of SNodes, thus facilitating the use of more spectrum efficient transmission configurations along fewer hops. However, the gain in spectrum saving by increasing SN LNR is much higher for high BSR

requirements. The additional gain is due to the use of three or more disjoint SPaths for mapping a VLink, thanks to the higher path diversity of a densely connected SN.

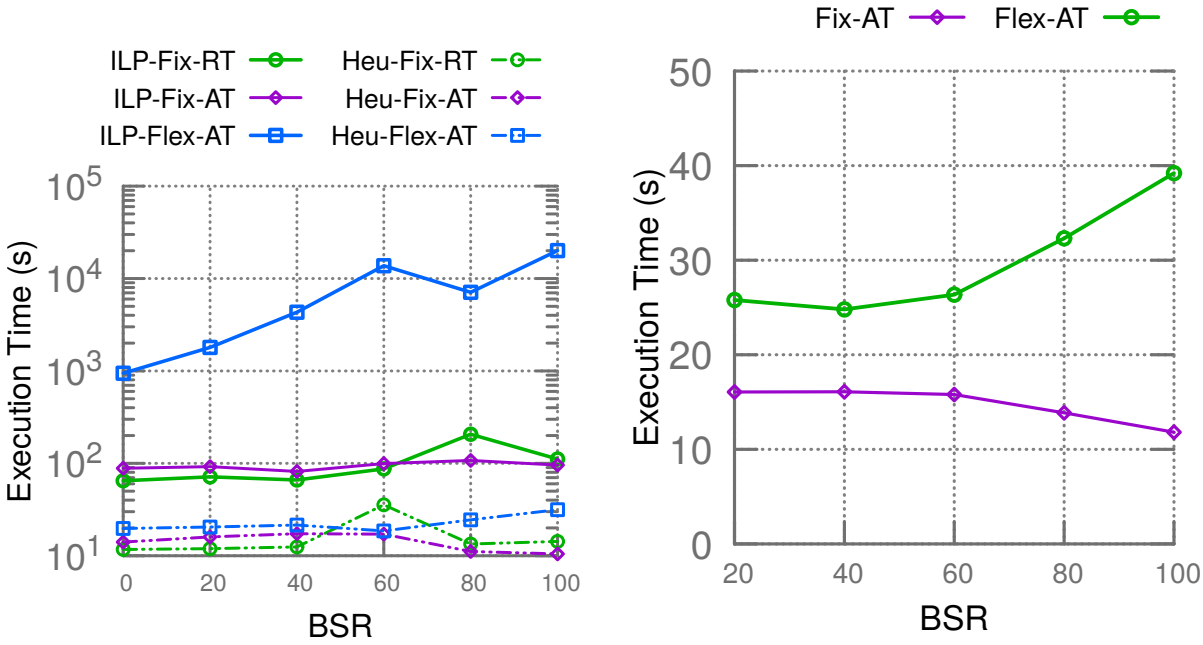
Finally, Fig. 5.5(b) demonstrates the benefit of our splitting model that enables *Flex-AT* to save 3%–4% SSU compared to a case in which splitting is allowed over different SPaths, but not over the same SPath on different spectrum segments (*Flex-AT-NoSplit-onSamePath*, e.g., the splitting model of [66]). Note that *NoSplit-onSamePath* model renders problem instances with high BSR in *Fix-RT* and *Fix-AT* cases infeasible as spectrum fragmentation prohibits only one split per feasible SPath to occupy the required spectrum. In contrast, our model finds solutions in those cases by increasing number of splits per SPath even in fragmented situations as shown in Fig. 5.4(a) and Fig. 5.4(d). It is worth noting that our splitting model does not require any additional hardware investment, so the savings ($\sim$ 4%) by our splitting model shown in Fig. 5.5(b) are achieved at no additional cost.



(a) Cost ratio (Nobel Germany EON)

(b) SSU (Germany50 EON)

Figure 5.6: Performance of our Heuristic Algorithm



(a) Small Scale (Nobel Germany)

(b) Large Scale (Germany50)

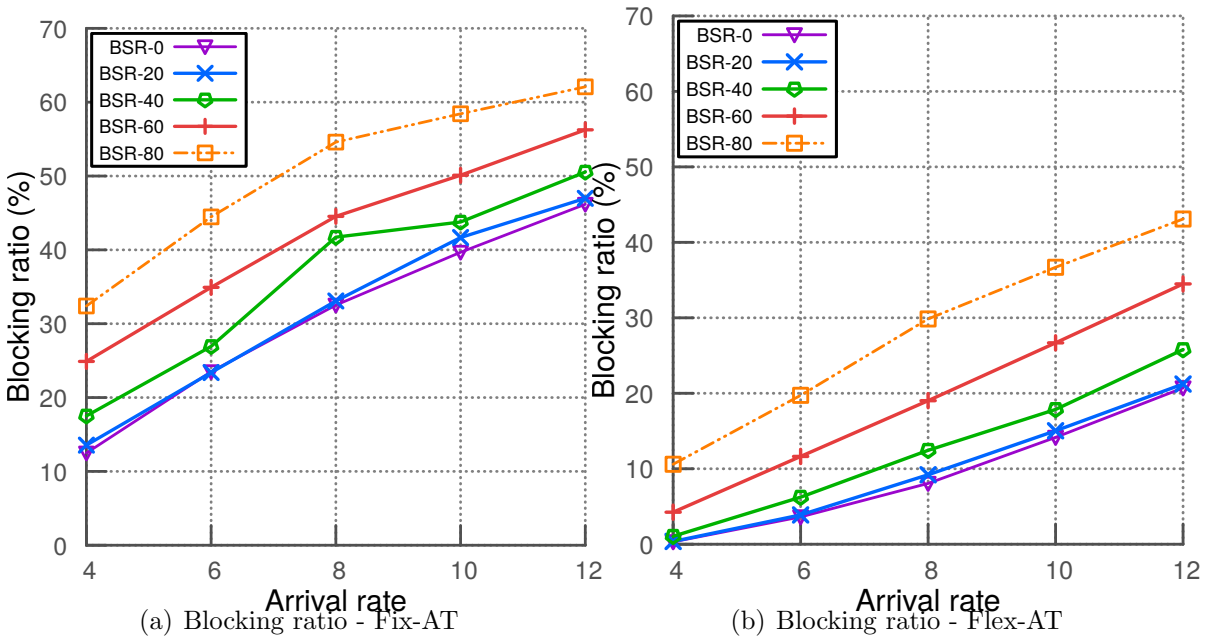
Figure 5.7: Execution Time

Performance of the Heuristic Algorithm

As our cost function (5.8) is dominated by SSU, the cost ratio between heuristic and ILP gives a measure of how much additional resources are allocated by the heuristic. We present the cumulative distribution function (CDF) of cost ratios in Fig. 5.6(a). Over all, the heuristic incurs 12%, 9%, and 7% additional cost on average compared to the ILP for *Fix-RT*, *Fix-AT*, and *Flex-AT*, respectively, while executing 2 or 3 orders of magnitude faster. Fig. 5.6(b) shows SSU incurred by the heuristic algorithm in Germany50 SN. Fig. 5.6(b) confirms the conclusions observed in Fig. 5.4, but with much larger and higher number of VNs, a bigger SN, and full 4THz spectrum resources on SLinks and by repeating our simulations over a much higher number of instances to achieve statistical confidence.

Scalability of the heuristic

We demonstrate the scalability of our heuristic algorithm compared to that of the ILP-based optimal solution by comparing their execution time in Fig. 5.7. Fig. 5.7(a) shows the



(a) Blocking ratio - Fix-AT (b) Blocking ratio - Flex-AT

Figure 5.8: Blocking ratio (Nobel Germany EON)

execution times of the ILP and heuristic algorithm on the small scale scenario. When we consider flex-grid spectrum allocation and flexible transponders, the ILP can take several hours to embed a VN with dedicated backup resources. With the increase in BSR, the solution space becomes larger hence the increase in execution time for ILP-Flex-AT. Compared to the ILP based solutions, the heuristic finds a solution several orders of magnitude faster. For the large scale scenario, the heuristic is able to find solutions in less than a minute (Fig. 5.7(b)) for the largest VN in our simulation. Again there is an increasing trend in execution time with increasing BSR for Flex-AT due to larger solution space. In contrast, the ILP fails to solve even a single problem instance in large scale scenario, limiting its scalability to only small scale problem instances.

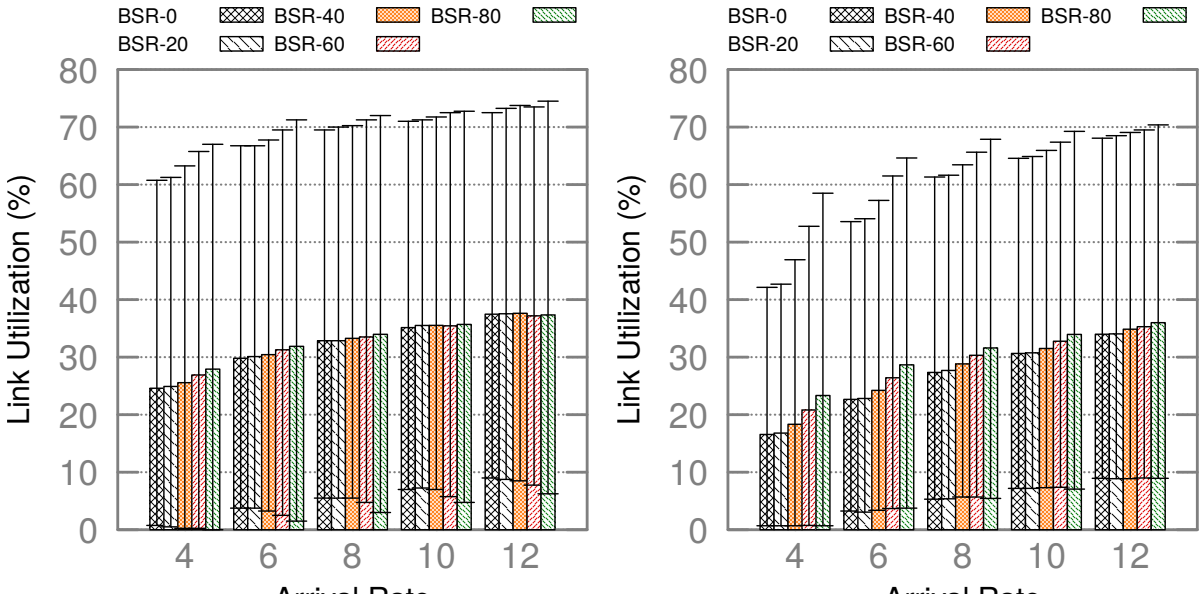
5.5.5 Steady state analysis

Blocking Ratio

Fig. 5.8 presents average blocking ratio for increasing arrival rate and different BSR considering both fixed- (Fig. 5.8(a)) and flex-grid (Fig. 5.8(b)) spectrum allocation using flexible

transponders. The results on blocking ratio reaffirms our previous finding that BSR less than 40% incurs very little protection overhead for the VNs. As a result, we see nearly identical blocking ratio for BSR 20% and BSR 0% (*i.e.*, no protection baseline scenario) in case of both fixed- and flex-grid EONs. Indeed, flex-grid EON has finer-grained resource allocation capabilities, resulting in significantly less blocking ratio for the same load levels compared to fixed-grid spectrum allocation. With increasing BSR, especially beyond BSR 40%, the protection overhead significantly increases for the same VN request, hence, the increasing gap between different BSRs in Fig. 5.8(a) and Fig. 5.8(b).

Substrate Link Utilization



(a) Link utilization - Fix-AT

(b) Link utilization - Flex-AT

Figure 5.9: Substrate Link Utilization (Nobel Germany EON)

Fig. 5.9 presents the average utilization of the substrate links during the course of simulation with 5th and 95th percentile errorbars. For a given load, *i.e.*, arrival rate, we see that link utilization does not significantly vary across different BSRs. From the first look, this result is counter-intuitive and does not seem to agree with the result from Fig. 5.8. One would expect a high blocking ratio resulting from the substrate links becoming saturated. In other words, link utilization for BSR 80% was expected to be significantly higher

compared to BSR 0%. However, even though a higher BSR results in lesser number of admitted VNs, it also results in higher protection overhead for the embedded VNs. As a consequence, for the same arrival rate with different BSR requirements, the higher BSR requirement will end up using more number of spectrum slots in the substrate links. This results in similar level of spectrum occupancy even with lower number of VNs admitted due to higher BSR requirement.

5.6 Conclusion

This chapter addresses a fundamental problem for the slicing of 5G transport networks, *i.e.*, reliable VN embedding with dedicated protection in an EON. To reduce resource overbuild of dedicated protection, we exploit bandwidth squeezing and VLink demand splitting over multiple SPaths, while leveraging the flexibility offered by an EON. Our novel splitting model not only provides the opportunity to split a VLink demand across multiple SPaths but also across multiple spectrum segments of an SPath. We present an ILP formulation to solve the problem optimally, and a heuristic solution to address the computational complexity of the ILP. Our simulations on realistic network topologies show that bandwidth squeezing and demand splitting allow to significantly reduce spectrum usage for providing dedicated protection, especially in the case of fully-flexible EON. Our evaluation also shows that the opportunity to have multiple splits over the same path allows to save additional spectrum compared to a model that does not allow splitting on the same path, and our proposed heuristic performs close to the optimal solution. For instance, multiple splits enable us to guarantee a BSR 40% with only 10% additional spectrum resources. This translates into similar VN blocking ratio up to 40% BSR. Our discrete event simulation also shows that similar spectrum occupancy is achieved on EON links irrespective of BSR requirement for a given VN arrival rate. However, spectrum occupancy increases with the increase in VN arrival rate.

Chapter 6

Conclusion and Future Research

6.1 Thesis Summary

As infrastructure providers are rolling out virtual networks as services, they are facing a number of challenges. One of the major challenges is to ensure virtual network survivability against failures in large physical infrastructures. Existing virtual network survivability literature fails to address the complexity and scale of the problem, resulting in over-simplified and hence impractical solutions. In this dissertation, we focused on the problem of Survivable Virtual Network Embedding (SVNE) with bandwidth and reliability guarantees for transport networks. We introduced four different survivability models to realize the vision of managing failures at the virtual network level as opposed to the physical network level. The contributions of the thesis are summarized below.

In chapter 2, we solved the *CoViNE* problem by ensuring VN connectivity in the presence of multiple substrate link failures. Specifically, we presented an ILP formulation, *CoViNE-opt*, that jointly solves the three sub-problems of *CoViNE*, namely, VN augmentation, computation of disjointness constraints, and VN embedding. To address the intractability of *CoViNE-opt*, we presented two sequential solutions *CoViNE-ILP* and *CoViNE-fast*. In contrast to the state-of-the-art, our solutions are general in their ability to handle multiple substrate link failures for arbitrary VN topologies. Our evaluation results demonstrated that *CoViNE-ILP* very closely approximates *CoViNE-opt*, and can be used as a baseline to compare heuristic algorithms. In contrast, *CoViNE-fast* scales to large topologies at the cost of provisioning about 16% additional resources compared to *CoViNE-ILP*, while executing several orders of magnitude faster for the same problem instances. Evaluation results also show that *CoViNE* for single and double link failure

scenarios require on average $\sim$ 24% and $\sim$ 66% extra resources than a VN embedding strategy that does not guarantee any connectivity. We also shown that VN connectivity can be leveraged to restore higher priority traffic in the presence of multiple substrate link failures.

Chapter 3 presented a new variant of the SVNE problem that requires a VN to be augmented with sufficient spare backup capacity and embedded on the physical network to ensure survivability against multiple substrate link failures. For this joint optimization problem, we have formulated the optimal solution as a QIP and transformed the QIP into an ILP, called *Opt-ILP*. We presented simplified ILP formulations to solve special cases of the joint optimization problem and mathematically analyzed the impact of physical network connectivity on the level of spare capacity sharing. We also proposed a heuristic algorithm to tackle the computational complexity of the ILP formulations. The heuristic algorithm provides a generic framework to survive multiple link failures and optimizes spare capacity for single and double link failure scenarios. Simulation results show that ILP formulations for the two special cases can provide upper and approximately lower bounds of the solution range. Moreover, the heuristic allocates $\sim$21% additional resources compared to the approximated lower bound, while executing several orders of magnitude faster. Large scale simulations of the heuristic algorithm shows that protection against double link failures comes at the cost of nearly two times the resources incurred by protection against single link failures. A quantitative comparison between the SVNE with VN level protection and the traditional SVNE with SN level protection reveals that the former can save $\sim$33% resources on average, compared to those required by the later.

In chapter 4, we addressed the problem of generalized recovery of a batch of affected VNs, resulting from a single substrate node failure. We formulated the problem as an ILP, *Opt-ReNoVatE* and presented an efficient heuristic algorithm, *Fast-ReNoVatE*, to tackle computational complexity. We have evaluated *Fast-ReNoVatE*, *Opt-ReNoVatE*, and a state-of-the-art solution, *Dyn-Recovery* in both small and large scale networks. Evaluation results demonstrate that *Fast-ReNoVatE* can recover $\sim$ 6% more VLinks than *Dyn-Recovery* and $\sim$ 3% less VLinks than *Opt-ReNoVatE* in high utilization scenarios. In terms of scalability, *Fast-ReNoVatE* is several orders of magnitude faster than *Opt-ReNoVatE*, and has comparable performance with *Dyn-Recovery*. In large scale networks, we have compared *Fast-ReNoVatE* with *Dyn-Recovery* and a baseline case of infinite bandwidth SN. The results demonstrated that *Fast-ReNoVatE* is able to recover $\sim$ 99% of the failed VNs if the SN has adequate residual capacity, and has similar timing performance to *Dyn-Recovery*. Furthermore, we investigated two variants of our recovery scheme, namely, fair recovery model (FRM) and priority-based recovery model (PRM). Our evaluation results suggest that FRM-based solutions fail to take into account variety of recovery require-

ments. In contrast, PRM-based solutions can prioritize the affected VNs based on SLA requirements, impacts of failure or profits, and adhere to that priority during recovery.

Chapter 5 discussed a fundamental problem for the slicing of 5G transport networks, *i.e.*, reliable VN embedding with dedicated protection in an EON. To reduce resource overbuild of dedicated protection, we exploit bandwidth squeezing and VLink demand splitting over multiple SPaths, while leveraging the flexibility offered by an EON. Our novel splitting model not only provides the opportunity to split a VLink demand across multiple SPaths but also across multiple spectrum segments of an SPath. We presented an ILP formulation to solve the problem optimally, and a heuristic solution to address the computational complexity of the ILP. Our simulations on realistic network topologies showed that bandwidth squeezing and demand splitting allow to significantly reduce spectrum usage for providing dedicated protection, especially in the case of fully-flexible EON. Our evaluation also showed that the opportunity to have multiple splits over the same path allows to save additional spectrum compared to a model that does not allow splitting on the same path, and our proposed heuristic performs close to the optimal solution. For instance, multiple splits enable us to guarantee a BSR of 40% with only 10% additional spectrum resources. This translates into similar VN blocking ratio up to 40% BSR. Our discrete event simulation also showed that similar spectrum occupancy is achieved on EON links irrespective of BSR requirement for a given VN arrival rate. However, spectrum occupancy increases with the increase in VN arrival rate.

In this dissertation, we have addressed four key research challenges in survivable network virtualization and proposed efficient solutions to solve them. We have also demonstrated the superiority of our proposed solutions through extensive simulations using realistic scenarios. However, there are issues that need further explorations. The next section presents some of these future research directions.

6.2 Future Research Directions

Survivability in 5G Network Slices. A 5G network slice spans multiple network segments, each of which can have different technological and physical constraints. For instance, the access network may be deployed as a passive optical network, which has limited bandwidth and scalability, to minimize cost and energy [101]. In contrast, the core transport network may not have any issues in terms of the capacity or scalability, thanks to elastic optical technologies [90]. However, the core transport may have higher cost and energy footprint due to long geographical distances and more complex optical devices. Similar trade-offs exist between edge and central DCs in terms of processing capacity, cost, and

energy consumption. Therefore, it will be impractical to provision a slice with a homogeneous survivability model for all network segments. A potential research direction is to develop mechanisms that translate a slice reliability requirement into sub-requirements for each segment the slice spans and select appropriate survivability mechanism (*e.g.*, dedicated protection with BSR, *CoViNE*, and *ReNoVatE*) for each segment. Such survivability decisions can be guided by predictive models for network equipments' failure and availability estimation with temporal and spatial considerations.

Adaptive Survivability. While an initial survivability model is effective for certain period of time, the initial model and its resource allocation may become obsolete later due to failure occurrences and network technology evolution [73]. For example, a network slice with dedicated backup protection can suffer if a primary resource fails first, followed by a failure in its backup route before the primary resource is repaired. To avoid such situations, the InP must continuously monitor its network elements and take proactive actions when observing any performance degradation (also called soft failure) or early detection of malfunction. In addition, the InP can estimate the likelihood of failures of the elements of a physical network and adapt the initial survivability model accordingly. An example of such adaptation can be to proactively migrate the slice away from the faulty element. An interesting research direction in this regard is to develop predictive models to infer future failure events based on current network state and any alarm that may have generated due to the malfunction. Another challenge is to schedule the survivability adaptation in a way that causes minimal to no disruption to existing traffic in the network slice.

Fast Failure Repair. Even with predictive models and adaptive survivability some failures, such as fiber cuts and device burns are inevitable. Ability of an InP to quickly repair a failure is crucial to keep the network operational. Failure repair involves three steps: failure detection, localization, and identification. The goal of failure detection is to trigger an alert after the failure has occured. Once detected, the failed element (e.g., the node or link responsible for the failure) must be localized in the network to narrow down the cause of failure. Even after localization, it might still be complex to understand the exact cause of the failure (e.g., inside a network node, the degradation can be due to misconfiguration or malfunction). To speed up the failure repair process, all the three steps of failure repair should be automated. An interesting avenue of research will be to develop models and algorithms for automated failure detection, localization, and identification in live networks. These models will decrease the mean time to repair of failure events, thus improving the availability of a network slice.

References

[1] The cost of downtime. [online]. . http://blogs.gartner.com/andrew-lerner/2014/07/16/the-cost-of-downtime/.

[2] Google IPv6 adopttion statistics. [online]. . https://www.google.com/intl/en/ipv6/statistics.html#tab=ipv6-adoption.

[3] International working group on cloud computing resiliency. [online]. https://iwgcr.files.wordpress.com/2012/06/iwgcr-paris-ranking-001-en1.pdf.

[4] ITU-T G.694.1. spectral grids for WDM applications: DWDM frequency grid.[online]. https://www.itu.int/rec/t-rec-g.694.1/en.

[5] Menger's theorem. [online]. http://math.fau.edu/locke/menger.htm.

[6] OpenFlow-enabled T-SDN. [online] . https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-of-enabled-transport-sdn.pdf.

[7] SNDlib data repository. [online]. http://sndlib.zib.de/home.action.

[8] Optical internetworking forum - flex ethernet implementation agreement 1.1, June 2017.

[9] Meeting 5G transport requirements with FlexE. White paper, 2018.

[10] Ibrahim Afolabi, Tarik Taleb, Konstantinos Samdanis, Adlen Ksentini, and Hannu Flinck. Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. *IEEE Communications Surveys & Tutorials*, 20(3):2429–2453, 2018.

[11] Sharad Agarwal, John Dunagan, Navendu Jain, Stefan Saroiu, Alec Wolman, and Habinder Bhogan.

[12] Slavisa Aleksic. Towards fifth-generation (5G) optical transport networks. In *Proc. of ICTON*, pages 1–4, 2015.

[13] Rodolfo Alvizu, Guido Maier, Navin Kukreja, Achille Pattavina, Roberto Morro, Alessandro Capello, and Carlo Cavazzoni. Comprehensive survey on t-sdn: Software-defined networking for transport networks. *IEEE Communications Surveys & Tutorials*, 19(4):2232–2283, 2017.

[14] Norberto Amaya, Georgios Zervas, and Dimitra Simeonidou. Introducing node architecture flexibility for elastic optical networks. *IEEE/OSA Journal of Opt. Commun. and Netw.*, 5(6):593–608, 2013.

[15] Aris Anagnostopoulos et al. A mazing 2+ ε approximation for unsplittable flow on a path. In *ACM SODA*, pages 26–41, 2014.

[16] K. D. R. Assis, S. Peng, R. C. Almeida, H. Waldman, A. Hammad, A. F. Santos, and D. Simeonidou. Network virtualization over elastic optical networks with different protection schemes. *IEEE/OSA Journal of Opt. Commun. and Netw.*, 8(4):272–281, Apr 2016.

[17] Rachna Asthana, Yatindra Nath Singh, and Wayne D Grover. p-cycles: An overview. *IEEE communications surveys & tutorials*, 12(1), 2010.

[18] Sara Ayoubi, Chadi Assi, Yiheng Chen, Tarek Khalifa, and Khaled Bashir Shaban. Restoration methods for cloud multicast virtual networks. *Journal of Network and Computer Applications*, 78:180–190, 2017.

[19] Sara Ayoubi, Chadi Assi, Lata Narayanan, and Khaled Shaban. Optimal polynomial time algorithm for restoring multicast cloud services. *IEEE Communications Letters*, 20(8):1543–1546, Aug 2016.

[20] Sara Ayoubi, Yiheng Chen, and Chadi Assi. Towards promoting backup-sharing in survivable virtual network design. *IEEE/ACM Transactions on Networking*, 24(5):3218–3231, 2016.

[21] Malti Baghel, Shikha Agrawal, and Sanjay Silakari. Survey of metaheuristic algorithms for combinatorial optimization. *Int. J. of Computer Applications*, 58(19), 2012.

[22] Nikhil Bansal, Ranjita Bhagwan, Navendu Jain, Yoonho Park, Deepak Turaga, and Chitra Venkatramani. Towards optimal resource allocation in partial fault-tolerant applications. In *IEEE INFOCOM*, 2008.

[23] Isil Burcu Barla, Dominic A Schupke, Marco Hoffmann, and Georg Carle. Optimal design of virtual networks for resilient cloud services. In *Design of Reliable Communication Networks (DRCN), 2013 9th International Conference on the*, pages 218–225. IEEE, 2013.

[24] Greg Bernstein, Diego Caviglia, Richard Rabbat, and Huub Van Helvoort. Vcat-lcas in a clamshell. *IEEE Commun. Mag.*, 44(5):34–36, 2006.

[25] LU Bo, Tao Huang, Xiao-chuan SUN, Jian-ya CHEN, and Yun-jie LIU. Dynamic Recovery for Survivable Virtual Network Embedding. *The Journal of China Universities of Posts and Telecommunications*, 21:77–84, Jun 2014.

[26] Peter Bodík, Ishai Menache, Mosharaf Chowdhury, Pradeepkumar Mani, David A Maltz, and Ion Stoica. Surviving failures in bandwidth-constrained datacenters. In *ACM SIGCOMM*, pages 431–442, 2012.

[27] Paul Bonsma et al. A constant-factor approximation algorithm for unsplittable flow on paths. *SIAM Journal on Computing*, 43(2):767–799, 2014.

[28] Raouf Boutaba, Nashid Shahriar, and Siavash Fathi. Elastic optical networking for 5G transport. *Journal of Netw. and Syst. Man.*, 25(4):819–847, 2017.

[29] Richard A Brualdi. Introductory combinatorics. *New York*, 3, 1992.

[30] Anliang Cai, Jun Guo, Rongping Lin, Gangxiang Shen, and Moshe Zukerman. Multicast routing and distance-adaptive spectrum allocation in elastic optical networks with shared protection. *IEEE/OSA Journal of Lightwave Tech.*, 34(17):4076–4088, 2016.

[31] Mao-cheng Cai. The maximal size of graphs with at most k edge-disjoint paths connecting any two adjacent vertices. *Discrete Mathematics*, 85(1):43–52, 1990.

[32] Zhiping Cai, Fang Liu, Nong Xiao, Qiang Liu, and Zhiying Wang. Virtual network embedding for evolving networks. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5. IEEE, 2010.

[33] Jorge Carapinha and Javier Jiménez. Network virtualization: a view from the bottom. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pages 73–80. ACM, 2009.

[34] Alberto Castro, Luis Velasco, Jaume Comellas, and Gabriel Junyent. On the benefits of multi-path recovery in flexgrid optical networks. *Photonic Netw. Commun.*, 28(3):251–263, 2014.

[35] Alberto Castro, Luis Velasco, Marc Ruiz, and Jaume Comellas. Single-path provisioning with multi-path recovery in flexgrid optical networks. In *Proc. of UMTCS*, pages 745–751, 2012.

[36] Eranda Cela. *The quadratic assignment problem: theory and algorithms*, volume 1. Springer Science & Business Media, 2013.

[37] Xiaolin Chang, Jogesh K Muppala, Bin Wang, Jiqiang Liu, and Longmei Sun. Migration cost aware virtual network re-embedding in presence of resource failures. In *Networks (ICON), 2012 18th IEEE International Conference on*, pages 24–29. IEEE, 2012.

[38] Bijoy Chand Chatterjee, Nityananda Sarma, and Eiji Oki. Routing and spectrum allocation in elastic optical networks: A tutorial. *IEEE Communications Surveys & Tutorials*, 17(3):1776–1800, 2015.

[39] Bowen Chen, Jie Zhang, Weisheng Xie, Jason P Jue, Yongli Zhao, and Gangxiang Shen. Cost-effective survivable virtual optical network mapping in flexible bandwidth optical networks. *IEEE/OSA Journal of Lightwave Tech.*, 34(10):2398–2412, 2016.

[40] Qingyun Chen et al. A survivable virtual network embedding scheme based on load balancing and reconfiguration. In *IEEE/IFIP NOMS*, pages 1–7, 2014.

[41] Xiaoliang Chen, Fan Ji, and Zuqing Zhu. Service availability oriented p-cycle protection design in elastic optical networks. *IEEE/OSA Journal of Opt. Commun. and Netw.*, 6(10):901–910, 2014.

[42] Xiaoliang Chen, Massimo Tornatore, Shilin Zhu, Fan Ji, Wenshuang Zhou, Cen Chen, Daoyun Hu, Liu Jiang, and Zuqing Zhu. Flexible availability-aware differentiated protection in software-defined elastic optical networks. *IEEE/OSA Journal of Lightwave Tech.*, 33(18):3872–3882, 2015.

[43] Yang Chen, Jianxin Li, Tianyu Wo, Chunming Hu, and Wantao Liu. Resilient virtual network service provision in network virtualization environments. In *Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on*, pages 51–58. IEEE, 2010.

[44] Mosharaf Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Trans. on Netw.*, 20(1):206–219, 2012.

[45] NM Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862–876, 2010.

[46] NM Mosharaf Kabir Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. Virtual network embedding with coordinated node and link mapping. In *INFOCOM 2009, IEEE*, pages 783–791. IEEE, 2009.

[47] Shihabur Rahman Chowdhury, Reaz Ahmed, Md Mashrur Alam Khan, Nashid Shahriar, Raouf Boutaba, Jeebak Mitra, and Feng Zeng. Dedicated protection for survivable virtual network embedding. *IEEE Transactions on Network and Service Management*, 13(4):913–926, 2016.

[48] Shihabur Rahman Chowdhury, Reaz Ahmed, Md Mashrur Alam Khan, Nashid Shahriar, Raouf Boutaba, Jeebak Mitra, and Feng Zeng. Protecting virtual networks with drone. In *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, pages 78–86. IEEE, 2016.

[49] Shihabur Rahman Chowdhury, Reaz Ahmed, Nashid Shahriar, Aimal Khan, Raouf Boutaba, Jeebak Mitra, and Liu Liu. Revine: Reallocation of virtual network embedding to eliminate substrate bottlenecks. In *IEEE/IFIP Integrated Network Management Symposium (IM)*, 2017.

[50] Shihabur Rahman Chowdhury, Sara Ayoubi, Reaz Ahmed, Nashid Shahriar, Raouf Boutaba, Jeebak Mitra, and Liu Liu. Multi-layer virtual network embedding. *IEEE Trans. on Netw. and Serv. Man.*, 15(3):1132–1145, 2018.

[51] Nicos Christofides. An algorithm for the chromatic number of a graph. *The Computer Journal*, 14(1):38–39, 1971.

[52] Piet Demeester, Michael Gryseels, Achim Autenrieth, Carlo Brianza, Laura Castagna, Giulio Signorelli, Roberto Clemenfe, Mauro Ravera, Andrzej Jajszczyk,

Dariusz Janukowicz, et al. Resilience in multilayer networks. *IEEE Communications Magazine*, 37(8):70–76, 1999.

[53] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[54] Yefim Dinitz et al. On the single-source unsplittable flow problem. *Combinatorica*, 19(1):17–41, 1999.

[55] D Anthony Dunn, Wayne D Grover, and Mike H MacGregor. Comparison of k-shortest paths and maximum flow routing for network facility restoration. *IEEE Journal on Selected Areas in Communications*, 12(1):88–99, Jan 1994.

[56] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, April 1972.

[57] Salah Eddine Elayoubi, Sana Ben Jemaa, Zwi Altman, and Ana Galindo-Serrano. 5G RAN slicing for verticals: Enablers and challenges. *IEEE Communications Magazine*, 57(1):28–34, 2019.

[58] Shimon Even et al. On the complexity of time table and multi-commodity flow problems. In *IEEE FOCS*, pages 184–193, 1975.

[59] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann De Meer, and Xavier Hesselbach. Virtual Network Embedding: A Survey. *IEEE CST*, Feb 2013.

[60] Xenofon Foukas et al. Network slicing in 5g: Survey and challenges. *IEEE Communications Magazine*, 55(5):94–100, 2017.

[61] Zachary Friggstad et al. On linear programming relaxations for unsplittable flow in trees. In *LIPIcs-Leibniz Int. Proc. in Informatics*, volume 40, 2015.

[62] Ornan Gerstel and Galen Sasaki. Quality of protection (qop): a quantitative unifying paradigm to protection service grades. *Optical Networks Magazine*, 3(3):40–49, 2002.

[63] Abdulaziz M Ghaleb, Tarek Khalifa, Sara Ayoubi, and Khaled Bashir Shaban. Surviving link failures in multicast vn embedded applications. In *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, pages 645–651. IEEE, 2016.

[64] Abdulaziz M Ghaleb, Tarek Khalifa, Sara Ayoubi, Khaled Bashir Shaban, and Chadi Assi. Surviving multiple failures in multicast virtual networks with virtual machines migration. *IEEE Transactions on Network and Service Management*, 13(4):899–912, 2016.

[65] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 350–361. ACM, 2011.

[66] Róża Goścień, Krzysztof Walkowiak, and Massimo Tornatore. Survivable multipath routing of anycast and unicast traffic in elastic optical networks. *IEEE/OSA Journal of Opt. Commun. and Netw.*, 8(6):343–355, 2016.

[67] Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. Evolve or die: High-availability design principles drawn from googles network infrastructure. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*, pages 58–72. ACM, 2016.

[68] Steven Gringeri, Bert Basch, Vishnu Shukla, Roman Egorov, and Tiejun J Xia. Flexible architectures for optical transport nodes and networks. *IEEE Commun. Mag.*, 48(7):40–50, 2010.

[69] W Grover. Distributed restoration of the transport network. *IEE TELECOMMUNICATIONS SERIES*, 30:337–337, 1994.

[70] Bingli Guo, Chunming Qiao, Jianping Wang, Hongfang Yu, Yongxia Zuo, Juhao Li, Zhangyuan Chen, and Yongqi He. Survivable virtual network design and embedding to survive a facility node failure. *Lightwave Technology, Journal of*, 32(3):483–493, 2014.

[71] Tao Guo, Ning Wang, Klaus Moessner, and Rahim Tafazolli. Shared Backup Network Provision for Virtual Network Embedding. In *IEEE ICC*, pages 1–5, Kyoto, Japan, Jun 2011.

[72] Ashish Gupta and Jeff Shute. High-availability at massive scale: Building google's data infrastructure for ads. 2015.

[73] Mohammad Hadi, Mohammad Reza Pakravan, and Erik Agrell. Dynamic resource allocation in metro elastic optical networks using lyapunov drift optimization. *Journal of Opt. Commun. & Netw.*, 11(6):250–259, 2019.

[74] Aun Haider and Richard J Harris. Recovery techniques in next generation networks. *IEEE Communications Surveys and Tutorials*, 9(1-4):2–17, 2007.

[75] Oded Hauser, Murali Kodialam, and TV Lakshman. Capacity design of fast path restorable optical networks. In *Proc. of IEEE INFOCOM*, pages 817–826, 2002.

[76] Wensheng He et al. Capacity optimization for surviving double-link failures in mesh-restorable optical networks. *Photonic Network Communications*, 9(1):99–111, 2005.

[77] Wensheng He and Arun K Somani. Path-based protection for surviving double-link failures in mesh-restorable optical networks. In *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*, volume 5, pages 2558–2563. IEEE, 2003.

[78] Sandra Herker, Ashiq Khan, and Xueli An. Survey on survivable virtual network embedding problem and solutions. In *International Conference on Networking and Services, ICNS*, 2013.

[79] Ali Hmaity et al. Survivable virtual network mapping to provide content connectivity against double-link failures. In *IEEE DRCN*, pages 160–166, 2016.

[80] Ines Houidi et al. Adaptive virtual network provisioning. In *ACM SIGCOMM VISA workshop*, pages 41–48, 2010.

[81] Qian Hu, Yang Wang, and Xiaojun Cao. Location-constrained survivable network virtualization. In *Sarnoff Symposium (SARNOFF), 2012 35th IEEE*, pages 1–5. IEEE, 2012.

[82] Qian Hu, Yang Wang, and Xiaojun Cao. Survivable network virtualization for single facility node failure: A network flow perspective. *Optical Switching and Networking*, 10(4):406–415, 2013.

[83] Sheng Huang, Charles U. Martel, and Biswanath Mukherjee. Survivable multipath provisioning with differential delay constraint in telecom mesh networks. *IEEE/ACM Trans. on Netw.*, 19(3):657–669, June 2011.

[84] Farabi Iqbal and Fernando A Kuipers. Disjoint paths in networks. *Wiley Encyclopedia of Electrical and Electronics Engineering*, 2015.

[85] Brigitte Jaumard and Hai Anh Hoang. Design and dimensioning of logical survivable topologies against multiple failures. *IEEE/OSA Journal of Optical Communications and Networking*, 5(1):23–36, 2013.

[86] Muhammad Javed et al. Lightpaths routing for single link failure survivability in ip-over-wdm networks. *Journal of Communications and Networks*, 9(4):394–401, 2007.

[87] Fan Ji, Xiaoliang Chen, Wei Lu, Joel JPC Rodrigues, and Zuqing Zhu. Dynamic p-cycle protection in spectrum-sliced elastic optical networks. *IEEE/OSA Journal of Lightwave Tech.*, 32(6):1190–1199, 2014.

[88] Huihui Jiang, Long Gong, and ZW Zuqing. Efficient joint approaches for location-constrained survivable virtual network embedding. In *Proc. of IEEE GLOBECOM*, pages 1810–1815, 2014.

[89] Huihui Jiang, Yixiang Wang, Long Gong, and Zuqing Zhu. Availability-aware survivable virtual network embedding in optical datacenter networks. *Journal of Optical Communications and Networking*, 7(12):1160–1171, 2015.

[90] Masahiko Jinno, Hidehiko Takara, Kazushige Yonenaga, and Akira Hirano. Virtualization in optical networks from network level to hardware level. *Journal of Optical Communications and Networking*, 5(10):A46–A56, 2013.

[91] Ahmed E. Kamal et al. Overlay protection against link failures using network coding. *IEEE/ACM Trans. Netw.*, 19(4):1071–1084, Aug 2011.

[92] Daniel Dao-Jun Kan et al. Lightpath routing and capacity assignment for survivable ip-over-wdm networks. In *IEEE DRCN*, 2009.

[93] Md Mashrur Alam Khan, Nashid Shahriar, Reaz Ahmed, and Raouf Boutaba. SiMPLE: Survivability in multi-path link embedding. In *Proc. of IEEE/ACM CNSM*, pages 210–218, 2015.

[94] Md Mashrur Alam Khan, Nashid Shahriar, Reaz Ahmed, and Raouf Boutaba. Multi-path link embedding for survivability in virtual networks. *IEEE Transactions on Network and Service Management*, 13(2):253–266, 2016.

[95] Bahare M Khorsandi, Federico Tonini, Elisabetta Amato, and Carla Raffaelli. Dedicated path protection for reliable network slice embedding based on functional splitting. In *Proc. of IEEE ICTON*, pages 1–4, 2019.

[96] Yuto Kishi, Nattapong Kitsuwan, Hiro Ito, Bijoy Chand Chatterjee, and Eiji Oki. Modulation-adaptive link-disjoint path selection model for 1+ 1 protected elastic optical networks. *IEEE Access*, 7:25422–25437, 2019.

[97] Jon Michael Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, Citeseer, 1996.

[98] Mirosław Klinkowski. A genetic algorithm for solving rsa problem in elastic optical networks with dedicated path protection. In *Proceedings of International Joint Conference CISIS'12-ICEUTE´ 12-SOCO´ 12 Special Sessions*, pages 167–176, 2013.

[99] Mirosław Klinkowski. An evolutionary algorithm approach for dedicated path protection problem in elastic optical networks. *Cybernetics and Systems*, 44(6-7):589–605, 2013.

[100] Simon Knight, Hung X Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[101] Glen Kramer. *What is Passive Optical Network?*, volume 17. McGraw-Hill, 2005.

[102] Eligijus Kubilinskas and Michal Pioro. Two design problems for the IP/MLPS over wdm networks. In *DRCN '05*, pages 241–248.

[103] Maciej Kurant and Patrick Thiran. Survivable mapping algorithm by ring trimming (smart) for large IP-over-WDM networks. In *BroadNets*, pages 44–53, Oct 2004.

[104] Maciej Kurant and Patrick Thiran. On survivable routing of mesh topologies in ip-over-wdm networks. In *IEEE INFOCOM*, pages 1106–1116, 2005.

[105] Maciej Kurant and Patrick Thiran. Survivable routing of mesh topologies in ip-over-wdm networks by recursive graph contraction. *IEEE Journal on Selected Areas in Communications*, 25(5), 2007.

[106] Eugene L Lawler. The quadratic assignment problem. *Management science*, 9(4):586–599, 1963.

[107] Kayi Lee et al. Cross-layer survivability in wdm-based networks. *IEEE/ACM Trans. on Netw.*, 19(4):1000–1013, Aug 2011.

[108] Tachun Lin et al. Logical topology survivability in ip-over-wdm networks: Survivable lightpath routing for maximum logical topology capacity and minimum spare capacity requirements. In *IEEE DRCN*, pages 1–8, 2011.

[109] Tachun Lin et al. Unified mathematical programming frameworks for survivable logical topology routing in ip-over-wdm optical networks. *IEEE/OSA Journal of Optical Communications and Networking*, 6(2):190–203, 2014.

[110] Chang Liu et al. A new survivable mapping problem in ip-over-wdm networks. *IEEE JSAC*, 25(3):25–34, Apr 2007.

[111] Menglin Liu, Massimo Tornatore, and Biswanath Mukherjee. Survivable traffic grooming in elastic optical networks—shared protection. *Journal of Lightwave Tech.*, 31(6):903–909, 2013.

[112] Xiao Liu et al. Disaster-prediction based virtual network mapping against multiple regional failures. In *IFIP/IEEE IM*, pages 371–378, 2015.

[113] Yu Liu et al. Approximating optimal spare capacity allocation by successive survivable routing. *IEEE/ACM Trans. on Netw.*, 13(1):198–211, 2005.

[114] Yu Liu, David Tipper, and Peerapon Siripongwutikorn. Approximating optimal spare capacity allocation by successive survivable routing. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 699–708. IEEE, 2001.

[115] Yu Liu et al. Spare capacity allocation in two-layer networks. *IEEE JSAC*, 25(5):974–986, 2007.

[116] K Makhijani, J Qin, R Ravindran, L Geng, L Qiang, S Peng, X de Foy, A Rahman, and A Galis. Network slicing use cases: Network customization and differentiated services. *draft-netslices-usecases-02 (Work in Progress)*, 2017.

[117] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, and Christophe Diot. Characterization of failures in an ip backbone. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2307–2317. IEEE, 2004.

[118] Sarang Bharadwaj Masti and Serugudi V Raghavan. Simulated annealing algorithm for virtual network reconfiguration. In *Next Generation Internet (NGI), 2012 8th EURO-NGI Conference on*, pages 95–102. IEEE, 2012.

[119] Márcio Melo et al. Virtual network mapping–an optimization problem. In *Mobile Networks and Management*, pages 187–200. Springer, 2012.

[120] Eytan Modiano et al. Survivable lightpath routing: a new approach to the design of wdm-based networks. *IEEE JSAC*, 2002.

[121] Navid Nikaein, Eryk Schiller, Romain Favraud, Kostas Katsalis, Donatos Stavropoulos, Islam Alyafawi, Zhongliang Zhao, Torsten Braun, and Thanasis Korakis. Network store: Exploring slicing in future 5g networks. In *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*, pages 8–13. ACM, 2015.

[122] Rodrigo R Oliveira et al. Dos-resilient virtual networks through multipath embedding and opportunistic recovery. In *ACM SAC '13*, pages 597–602.

[123] Rodrigo R Oliveira, Daniel S Marcon, Leonardo Richter Bays, Miguel C Neves, Luciana Salete Buriol, Luciano Paschoal Gaspary, and Marinho Pilla Barcellos. No more backups: Toward efficient embedding of survivable virtual networks. In *Communications (ICC), 2013 IEEE International Conference on*, pages 2128–2132. IEEE, 2013.

[124] Muhittin Oral and Ossama Kettani. A linearization procedure for quadratic and cubic mixed-integer problems. *Operations Research*, 40(1-supplement-1):S109–S116, 1992.

[125] Sebastian Orlowski, Roland Wessäly, Michal Pióro, and Artur Tomaszewski. Sndlib 1.0—survivable network design library. *Networks: An International Journal*, 55(3):276–286, 2010.

[126] Canhui Ou, Laxman H Sahasrabuddhe, Keyao Zhu, Charles U Martel, and Biswanath Mukherjee. Survivable virtual concatenation for data over sonet/sdh in optical transport networks. *IEEE/ACM Trans. on Netw.*, 14(1):218–231, 2006.

[127] Albert Pagès, Jordi Perelló, Salvatore Spadaro, and Jaume Comellas. Optimal route, spectrum, and modulation level assignment in split-spectrum-enabled dynamic elastic optical networks. *IEEE/OSA Journal of Opt. Commun. and Netw.*, 6(2):114–126, 2014.

[128] Mahsa Pourvali et al. Progressive recovery for network virtualization after large-scale disasters. In *IEEE ICNC*, pages 1–5, 2016.

[129] Md Golam Rabbani, Mohamed Faten Zhani, and Raouf Boutaba. On achieving high survivability in virtualized data centers. *IEICE Transactions on Communications*, 97(1):10–18, 2014.

[130] Prabhakar Raghavan and Clark D Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[131] Muntasir Raihan Rahman and Raouf Boutaba. SVNE: Survivable virtual network embedding algorithms for network virtualization. *IEEE Transactions on Network and Service Management*, 10(2):105–118, 2013.

[132] Muntasir Raihan Rahman et al. Survivable virtual network embedding. In *NETWORKING*. 2010.

[133] Senthil Ramamurthy and Biswanath Mukherjee. Survivable wdm mesh networks. part i-protection. In *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, volume 2, pages 744–751. IEEE, 1999.

[134] Sumathi Ramamurthy, Laxman Sahasrabuddhe, and Biswanath Mukherjee. Survivable wdm mesh networks. *IEEE/OSA Journal of Lightwave Tech.*, 21(4):870, 2003.

[135] Matthias Rost and Stefan Schmid. Charting the complexity landscape of virtual network embeddings. In *Proc. of IFIP Networking*, pages 55–63, 2018.

[136] Rajesh Roy and Biswanath Mukherjee. Degraded-service-aware multipath provisioning in telecom mesh networks. In *Proc. of IEEE/OSA OFC*, 2008.

[137] Lu Ruan and Nan Xiao. Survivable multipath routing and spectrum allocation in OFDM-based flexible optical networks. *IEEE/OSA Journal of Opt. Commun. and Netw.*, 5(3):172–182, 2013.

[138] Lu Ruan and Yanwei Zheng. Dynamic survivable multipath routing and spectrum allocation in ofdm-based flexible optical networks. *Journal of Opt. Commun. and Netw.*, 6(1):77–85, 2014.

[139] Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *Journal of the ACM (JACM)*, 23(3):555–565, 1976.

[140] Nashid Shahriar, Reaz Ahmed, Shihabur Rahman Chowdhury, Aimal Khan, Raouf Boutaba, and Jeebak Mitra. Generalized recovery from node failure in virtual network embedding. *IEEE Transactions on Network and Service Management*, 14(2):261–274, June 2017.

[141] Nashid Shahriar, Reaz Ahmed, Shihabur Rahman Chowdhury, Md Mashrur Alam Khan, Raouf Boutaba, Jeebak Mitra, and Feng Zeng. Connectivity-aware virtual network embedding. In *IFIP Networking Conference (IFIP Networking) and Workshops, 2016*, pages 46–54. IEEE, 2016.

[142] Nashid Shahriar, Reaz Ahmed, Shihabur Rahman Chowdhury, Md Mashrur Alam Khan, Raouf Boutaba, Jeebak Mitra, and Feng Zeng. Virtual network embedding

with guaranteed connectivity under multiple substrate link failures. *IEEE Trans. on Commun.*, 68(2):1025–1043, 2020.

[143] Nashid Shahriar, Reaz Ahmed, Aimal Khan, Shihabur Rahman Chowdhury, Raouf Boutaba, and Jeebak Mitra. Renovate: Recovery from node failure in virtual network embedding. In *IEEE/ACM/IFIP Conference on Network and Service Management (CNSM)*, 2016.

[144] Nashid Shahriar, Shihabur Rahman Chowdhury, Reaz Ahmed, Aimal Khan, Raouf Boutaba, Jeebak Mitra, and Liu Liu. Joint backup capacity allocation and embedding for survivable virtual networks. In *IFIP Networking Conference*, 2017.

[145] Nashid Shahriar, Shihabur Rahman Chowdhury, Reaz Ahmed, Aimal Khan, Siavash Fathi, Raouf Boutaba, Jeebak Mitra, and Liu Liu. Virtual network survivability through joint spare capacity allocation and embedding. *IEEE Journal on Selected Areas in Communication*, 36(3):502–518, 2018.

[146] Nashid Shahriar, Sepehr Taeb, Shihabur Rahman Chowdhury, Mubeen Zulfiqar, Massimo Tornatore, Raouf Boutaba, Jeebak Mitra, and Mahdi Hemmati. Reliable slicing of 5g transport networks with bandwidth squeezing and multi-path provisioning. *IEEE Transactions on Network and Service Management*, 2020.

[147] Nashid Shahriar, Sephr Taeb, Shihabur Rahman Chowdhury, Massimo Tornatore, Raouf Boutaba, Jeebak Mitra, and Mahdi Hemmati. Achieving a fully-flexible virtual network embedding in elastic optical networks. In *Proc. of IEEE INFOCOM*, pages 1756–1764, 2019.

[148] Nashid Shahriar, Sephr Taeb, Shihabur Rahman Chowdhury, Mubeen Zulfiqar, Massimo Tornatore, Raouf Boutaba, Jeebak Mitra, and Mahdi Hemmati. Reliable slicing of 5G transport networks with dedicated protection. In *Proc. of IEEE/ACM CNSM*, pages 1–9, 2019.

[149] Gangxiang Shen, Yue Wei, and Sanjay Bose. Optimal design for shared backup path protected elastic optical networks under single-link failure. *IEEE/OSA Journal of Opt. Commun. and Netw.*, 6(7):649–659, 2014.

[150] Milton A Soares and Edmundo RM Madeira. A multi-agent architecture for autonomic management of virtual networks. In *IEEE/IFIP NOMS*, pages 1183–1186, 2012.

[151] Y Sone, A Watanabe, W Imajuku, Y Tsukishima, B Kozicki, H Takara, and M Jinno. Bandwidth squeezed restoration in spectrum-sliced elastic optical path networks (slice). *IEEE/OSA Journal of Opt. Commun. and Netw.*, 3(3):223–233, 2011.

[152] Oussama Soualah, Ilhem Fajjari, Nadjib Aitsaadi, and Abdelhamid Mellouk. A batch approach for a survivable virtual network embedding based on monte-carlo tree search. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 36–43. IEEE, 2015.

[153] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson. Measuring isp topologies with rocketfuel. *IEEE/ACM Transactions on Networking (ToN)*, 12(1):2–16, 2004.

[154] Demetrios Stamatelakis and Wayne D Grover. IP layer restoration and network planning based on virtual protection cycles. *IEEE Journal on Selected Areas in Communications*, 18(10):1938–1949, Sep 2006.

[155] Sephr Taeb, Nashid Shahriar, Shihabur Rahman Chowdhury, Massimo Tornatore, Raouf Boutaba, Jeebak Mitra, and Mahdi Hemmati. Virtual network embedding with path-based latency guarantees in elastic optical networks. In *Proc. of IEEE ICNP*, pages 1–12, 2019.

[156] Krishnaiyan Thulasiraman et al. Logical topology augmentation for guaranteed survivability under multiple failures in ip-over-wdm optical networks. *Optical Switching and Networking*, 7(4):206–214, 2010.

[157] Krishnaiyan Thulasiraman et al. Primal meets dual: A generalized theory of logical topology survivability in ip-over-wdm optical networks. In *IEEE COMSNETS*, pages 1–10. IEEE, 2010.

[158] Krishnaiyan Thulasiraman, Muhammad S Javed, and Guoliang Xue. Circuits/cutsets duality and a unified algorithmic framework for survivable logical topology design in ip-over-wdm optical networks. In *INFOCOM*, pages 1026–1034. IEEE, 2009.

[159] Ajay Todimala et al. A scalable approach for survivable virtual topology routing in optical wdm networks. *IEEE JSAC*, 25(6):63–69, 2007.

[160] Phuong Nga Tran, Leonardo Casucci, and Andreas Timm-Giel. Optimal mapping of virtual networks considering reactive reconfiguration. In *Cloud Networking (CLOUDNET), 2012 IEEE 1st International Conference on*, pages 35–40. IEEE, 2012.

[161] Phuong Nga Tran and Andreas Timm-Giel. Reconfiguration of virtual network mapping considering service disruption. In *2013 IEEE International Conference on Communications (ICC)*, pages 3487–3492. IEEE, 2013.

[162] Jonathan S Turner and David E Taylor. Diversifying the internet. In *GLOBECOM'05. IEEE Global Telecommunications Conference, 2005.*, volume 2, pages 6–pp. IEEE, 2005.

[163] Krzysztof Walkowiak, Mirosław Klinkowski, Bartosz Rabiega, and Róża Goścień. Routing and spectrum allocation algorithms for elastic optical networks with dedicated path protection. *Elsevier Opt. Switching and Netw.*, 13:63–75, 2014.

[164] Chao Wang, Gangxiang Shen, and Sanjay Kumar Bose. Distance adaptive dynamic routing and spectrum allocation in elastic optical networks with shared backup path protection. *IEEE/OSA Journal of Lightwave Technology*, 33(14):2955–2964, 2015.

[165] Wei Wang, Yi Lin, Yongli Zhao, Guoying Zhang, Jie Zhang, Jianrui Han, Haoran Chen, Baogang Hou, Yuefeng Ji, and Liangjia Zong. First demonstration of virtual transport network services with multi-layer protection schemes over flexi-grid optical networks. *IEEE Communications Letters*, 20(2):260–263, 2016.

[166] Yang Wang, Xiaojun Cao, and Yi Pan. A study of the routing and spectrum allocation in spectrum-sliced elastic optical path networks. In *Proc. of IEEE INFOCOM*, pages 1503–1511, 2011.

[167] Yuyu Wang, Xin Li, Bingli Guo, Tao Gao, Wenzhe Li, and Shanguo Huang. Survivable virtual optical network mapping in elastic optical networks with shared backup path protection. In *Proc. of WOCC*, pages 1–4, 2016.

[168] Toshimasa Watanabe and Akira Nakamura. A minimum 3-connectivity augmentation of a graph. *Journal of Computer and System Sciences*, 46(1):91–128, 1993.

[169] Yue Wei, Kai Xu, Heming Zhao, and Gangxiang Shen. Applying p-cycle technique to elastic optical networks. In *Proc. of ONDM*, pages 1–6, 2014.

[170] Weisheng Xie, Jason P Jue, Qiong Zhang, Xi Wang, Qingya She, Paparao Palacharla, and Motoyoshi Sekiya. Survivable impairment-constrained virtual optical network mapping in flexible-grid optical networks. *IEEE/OSA Journal of Opt. Commun. and Netw.*, 6(11):1008–1017, 2014.

[171] Jielong Xu et al. Survivable virtual infrastructure mapping in virtualized data centers. In *IEEE CLOUD*, 2012.

[172] Hui Yang, Xiaoxu Zhu, Wei Bai, Yongli Zhao, Jie Zhang, Zhu Liu, Ziguan Zhou, and Qinghai Ou. Survivable von mapping with ambiguity similitude for differentiable maximum shared capacity in elastic optical networks. *Optical Fiber Technology*, 31:138–146, 2016.

[173] Zilong Ye et al. Survivable virtual infrastructure mapping with dedicated protection in transport software-defined networks [invited]. *JOCN*, 7(2):A183–A189, 2015.

[174] Wai-Leong Yeow, Cédric Westphal, and Ulas C Kozat. Designing and embedding reliable virtual infrastructures. *ACM SIGCOMM Computer Communication Review*, 41(2):57–64, 2011.

[175] Shan Yin, Shanguo Huang, Bingli Guo, Yu Zhou, Haibin Huang, Min Zhang, Yongli Zhao, Jie Zhang, and Wanyi Gu. Shared-protection survivable multipath scheme in flexible-grid optical networks against multiple failures. *IEEE/OSA Journal of Lightwave Tech.*, 35(2):201–211, 2016.

[176] Hongfang Yu et al. Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures. In *IEEE GLOBECOM*, pages 1–6, 2010.

[177] Hongfang Yu et al. Cost efficient design of survivable virtual infrastructure to recover from facility node failures. In *IEEE ICC*, pages 1–6, 2011.

[178] Guoying Zhang, Marc De Leenheer, Annalisa Morea, and Biswanath Mukherjee. A survey on OFDM-based elastic core optical networking. *IEEE Commun. Surveys Tuts.*, 15(1):65–87, 2012.

[179] Qi Zhang, Mohamed Faten Zhani, Maissa Jabri, and Raouf Boutaba. Venice: Reliable virtual data center embedding in clouds. In *INFOCOM, 2014 Proceedings IEEE*, pages 289–297. IEEE, 2014.

[180] Yongli Zhao, Bowen Chen, Jie Zhang, and Xinbo Wang. Energy efficiency with sliceable multi-flow transponders and elastic regenerators in survivable virtual optical networks. *IEEE Trans. on Commun.*, 64(6):2539–2550, 2016.

[181] Z. Zhou et al. Cross-layer network survivability under multiple cross-layer metrics. *IEEE/OSA J. of Optical Comm. & Net.*, 2015.

[182] Zhili Zhou et al. Novel survivable logical topology routing by logical protecting spanning trees in ip-over-wdm networks. *IEEE/ACM Transactions on Networking*, 25(3):1673–1685, 2017.

[183] Zhili Zhou et al. Survivable cloud network design against multiple failures through protecting spanning trees. *Journal of Lightwave Technology*, 35(2):288–298, 2017.

[184] Qiang Zhu et al. A hybrid reliable heuristic mapping method based on survivable virtual networks for network virtualization. *Discrete Dynamics in Nature and Society*, 2015, 2015.

[185] Yong Zhu and Mostafa H Ammar. Algorithms for assigning substrate network resources to virtual network components. In *IEEE INFOCOM*, 2006.