Joint Backup Capacity Allocation and Embedding for Survivable Virtual Networks

Nashid Shahriar*, Shihabur Rahman Chowdhury*, Reaz Ahmed*, Aimal Khan*, Raouf Boutaba*,
Jeebak Mitra†, and Liu Liu‡
*David R. Cheriton School of Computer Science, University of Waterloo
{nshahria | sr2chowdhury | r5ahmed | a273khan | rboutaba}@uwaterloo.ca
†Huawei Technologies Canada Research Center
jeebak.mitra@huawei.com
‡Huawei Technologies
liuliu1@huawei.com

***Abstract*—A key challenge in Network Virtualization is to efficiently map a virtual network (VN) on a substrate network (SN) while accounting for possible substrate failures. This is known as the Survivable Virtual Network Embedding (SVNE) problem. The state-of-the-art literature has studied the SVNE problem from infrastructure providers' (InPs) perspective, *i.e.*, provisioning backup resources in the SN. A rather unexplored solution spectrum is to augment the VN with sufficient spare backup capacity to survive substrate failures and embed the resulting VN accordingly. Such augmentation enables InPs to offload failure recovery decisions to the VN operator, thus, providing more flexible VN management. In this paper, we study the problem of jointly optimizing spare backup capacity allocation in a VN and embedding the VN to guarantee full bandwidth in the presence of single substrate link failure. We formulate the optimal solution to the joint optimization problem as a quadratic integer program that we transform into an integer linear program. We propose a heuristic algorithm to solve larger instances of the problem. Simulation results show that our heuristic allocates ∼21% extra resources compared to the optimal, while executing several orders of magnitude faster.**

I. Introduction

Infrastructure providers (InPs), such as data center network operators, Internet service providers, and transport network operators are leveraging Network Virtualization (NV) to offer slices of their networks to service providers (SPs) [1], [2]. NV enables InPs to better utilize their infrastructure, *i.e.*, substrate network (SN) and also to open new revenue streams. However, the benefits from NV come with additional resource management challenges, such as efficiently mapping the virtual nodes and links of a virtual network (VN) request onto substrate nodes and paths, respectively. This is known as the VN embedding (VNE) problem [3]. If a VNE solution does not take possible substrate failures into account, then such failures can result in degraded Quality of Service (QoS) for VNs, leading to Service Level Agreement (SLA) violations. A VN embedding that can survive substrate failures is known as the Survivable VNE (SVNE) [4], and has received significant attention in the research community [5]–[13].

The SVNE research literature focuses primarily on proactively (protection during embedding) or reactively (restoration after failure) provisioning backup resources in the SN, possibly disjoint from the VN's primary resources. A rather unexplored spectrum in SVNE is to augment a VN with adequate backup capacity and to embed the VN on an SN accordingly [14]. More recently, [7] has studied a weaker version of the SVNE problem that augments the VN topology to ensure connectivity under multiple substrate link failures. However, [7] does not guarantee the affected virtual links' bandwidth, and only allows the VN to operate with degraded QoS. In this paper, we focus on the problem of augmenting a VN with sufficient backup capacity and embedding the VN on an SN in a way that ensures survivability with guaranteed bandwidth under single substrate link failure.

The motivation for providing protection at the VN level comes from the shift of control to SPs, as anticipated in future Transport Software Defined Networks (T-SDNs) [15]. T-SDNs are the next generation of transport networks that leverage SDN technologies and promise to provide full fledged VN instead of traditional end-to-end connectivity to SPs [2], [16]. SPs can thus have more control over their virtual slice and deploy their own routing, traffic engineering, and failure recovery solutions. InPs can offload some of the failure recovery tasks to SPs by augmenting the VNs with sufficient spare capacity for backup and embedding the VNs using necessary disjoint paths in the SN. A recent study empirically evaluated the impact of providing survivability at the SN level, compared to that at the VN level, in a real testbed [15]. The study shows that: (i) providing survivability at the VN level has similar switching response time during a failure as compared to doing the same at the SN level, and (ii) VN level survivability can accommodate more VNs compared to doing the same at the SN, hence, is more profitable for InPs. The intuition behind the latter result is that InPs need to provision more disjoint resources when ensuring survivability at the SN level [15].

Another motivating application for providing survivability at the VN level is to ensure strong IP layer survivability in IP over Wavelength Division Multiplexed (WDM) networks. Strong survivability in this context means ensuring both connectivity and bandwidth at the IP layer during failures in either the IP or the WDM layer [17]. By equipping the IP layer with necessary backup resources needed to recover from a failure, part of the failure recovery tasks can be off-loaded to the IP

ISBN 978-3-901882-94-4

layer. However, a major difference between NV and IP-over-WDM is that IP routers are provisioned in fixed locations, whereas the virtual nodes are not assumed to have been already placed and the embedding algorithm determines their placement. Therefore, solutions for IP-over-WDM networks such as [17], [18] cannot be directly applied in NV context. As a matter of fact, the problem in NV context is more general than its instance in IP-over-WDM networks.

In this paper, we study the problem of jointly optimizing spare backup capacity allocation within a VN and embedding the VN on an SN to ensure survivability under single substrate link failure, while minimizing resource provisioning cost in the SN. We focus on the single link failure scenario since this is the most common case [19], [20]. A major challenge in solving the problem is to jointly optimize spare backup capacity allocation and survivable embedding. Spare capacity allocation and VN embedding, when performed independently of one another, may lead to suboptimal or infeasible solutions. Hence, we propose a joint optimization model to solve spare capacity allocation and VN embedding simultaneously. Specifically, we make the following contributions:

- We formulate the optimal solution to the joint optimization problem as a Quadratic Integer Program (QIP). We also present a transformation of the QIP into an Integer Linear Program (ILP).
- We propose a heuristic solution to tackle the computational complexity of the ILP-based optimal solution.
- We perform extensive simulations to evaluate our solutions. Simulation results show that our proposed heuristic allocates $\sim$21% extra resources compared to the optimal.

The rest of this paper is organized as follows. We present the related literature in § II and contrast our work with the state-of-the-art. In § III, we present the system model and problem statement followed by a discussion on how spare backup capacity can be allocated along a virtual link. Then, we present our QIP formulation for the joint optimization problem and the ILP transformation in § IV. We present the design of our heuristic in § V. The evaluation of our solutions are presented in § VI. Finally, we conclude with some future research directions in § VII.

II. Related Work

First, we discuss the state-of-the-art in SVNE in § II-A. Then, we present the works focused on providing protection at the VN level in § II-B. Finally, we contrast our approach with those from IP-over-WDM network survivability in § II-C.

A. SVNE with Protection at SN

Rahman *et al.*, were the first to address the SVNE problem by formulating the problem as a mixed integer linear program [4]. A number of subsequent research works have addressed different aspects of SVNE such as substrate node failure [5], [6], leveraging multi-path embedding [8], [9], shared backup protection [10], [11], and dedicated VN topology protection [12], [13]. However, these approaches address the SVNE problem from an InP's perspective, *i.e.*, the InP provisions backup resources in the SN, disjoint from the primary embedding. They do not explore the solution space where the VN can be augmented with sufficient resources to survive substrate failures and embed the VN accordingly.

B. SVNE with Protection at VN

Recently, an empirical study by Wang *et al.*, [15] compared different protection schemes for NV in T-SDNs. The results from [15] show that providing protection at the VN level can increase VN acceptance ratio. However, [15] performed a two step backup capacity allocation and embedding rather than jointly optimizing them. We studied a weaker version of the SVNE problem with VN level protection in [7]. This work proposed to embed a VN on an SN in such way that VN connectivity is ensured against multiple substrate link failures. However, it does not guarantee any bandwidth in case of a failure and only allow the VN to operate in a best effort manner. Barla *et al.*, proposed separate design models for cloud services that provide resiliency either at the VN or at the SN layer [14]. Their VN-based resiliency model employs dedicated backup consisting of additional virtual links to reach the recovery data center. In contrast, our model eliminates the need for changing the VN topology and uses spare capacity allocated to existing virtual links to survive link failures.

C. IP-over-WDM Network Survivability

A similar problem to our joint optimization has been studied in IP-over-WDM literature, namely, *Strongly Survivable Routing (SSR)*. The objective of the SSR is to ensure both connectivity and bandwidth guarantee at the IP layer during a failure in either IP or WDM layer. One approach for solving the SSR is to provision spare bandwidth at the IP layer to survive IP or WDM link failure [17], [18], [21]–[25]. However, these solutions do not jointly optimize spare bandwidth allocation and routing. Moreover, IP-over-WDM networks assume a fixed placement of IP routers in the network, whereas an SVNE algorithm needs to determine both VNode and VLink mappings. Therefore, solutions from IP-over-WDM cannot be directly applied to our joint optimization problem.

III. System Model and Background

We first present basic notations in § III-A and a formal statement of the problem in § III-B. We explain the concept of shared risk groups in § III-C. We then discuss how embedding affects spare capacity allocation on virtual links in § III-D. A glossary of notations used in the paper is provided in Table I.

A. Basic Notations

1) 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

TABLE I
NOTATION TABLE

$G = (V, E)$	Substrate Network (SN)
b_{uv}	Bandwidth capacity of SLink $(u, v) \in E$
C_{uv}	Cost of unit bandwidth on SLink $(u, v) \in E$
$\hat{G} = (\hat{V}, \hat{E})$	Virtual Network (VN)
$b_{\hat{u}\hat{v}}$	Bandwidth demand of VLink $(\hat{u}, \hat{v}) \in \hat{E}$
$L(\hat{u})$	Location constraint set for VNode $\hat{u} \in \hat{V}$
$\ell_{\hat{u}u} \in \{0, 1\}$	$\ell_{\hat{u}u} = 1$ if $u \in L(\hat{u})$, $u \in V, \hat{u} \in \hat{V}$
$\hat{P}_{\hat{u}\hat{v}}$	A VPath between $\hat{u}$ and $\hat{v}$, edge disjoint from $(\hat{u}, \hat{v})$
$P_{\hat{u}\hat{v}}$	An SPath representing the mapping of $(\hat{u}, \hat{v}) \in \hat{E}$
$S_{\hat{u}\hat{v}}$	Spare backup bandwidth allocated to $(\hat{u}, \hat{v}) \in \hat{E}$
$d_i \in D$	An SRG consisting of a set of VLinks from $\hat{E}$
$d_i^{\hat{u}\hat{v}}$	$d_i^{\hat{u}\hat{v}} = 1$ if $(\hat{u}, \hat{v}) \in \hat{E}$ belongs to SRG $d_i \in D$
$\hat{\mathcal{H}}_{\hat{u}\hat{v}} \subseteq \hat{E}$	Set of VLinks that have $(\hat{u}, \hat{v})$ in their backup VPaths
$z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}} \in \{0, 1\}$	$z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}} = 1$ if $(\hat{u}, \hat{v}) \in \hat{E}$ is on the backup VPath of $(\hat{x}, \hat{y}) \in \hat{E}$
$x_{uv}^{\hat{u}\hat{v}} \in \{0, 1\}$	$x_{uv}^{\hat{u}\hat{v}} = 1$ if $(u, v) \in E$ is on the embedded SPath for $(\hat{u}, \hat{v}) \in \hat{E}$
$y_{\hat{u}u} \in \{0, 1\}$	$y_{\hat{u}u} = 1$ if $\hat{u} \in \hat{V}$ is mapped to $u \in V$
$g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i) \in \{0, 1\}$	$g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i) = 0$ if $z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}} = 1$ and $d_i^{\hat{x}\hat{y}} = 1$
$q_{uv}^{\hat{u}\hat{v}} \in \{0, S_{\hat{u}\hat{v}}\}$	$q_{uv}^{\hat{u}\hat{v}} = S_{\hat{u}\hat{v}}$ if $x_{uv}^{\hat{u}\hat{v}} = 1$

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.

2) 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, $L = \{L(\hat{u}) | L(\hat{u}) \subseteq V, \forall \hat{u} \in \hat{V}\}$, 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 the VNs are 2-edge connected, *i.e.*, at least two edge disjoint paths exist between any two VNodes. 2-edge connectivity is a necessary condition to ensure that an edge disjoint backup virtual path always exists for each VLink $(\hat{u}, \hat{v}) \in \hat{E}$ [17].

B. Problem Statement

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

- For each VLink $(\hat{u}, \hat{v}) \in \hat{E}$, optimally allocate spare backup bandwidth along a non-empty path $\hat{P}_{\hat{u}\hat{v}}$ in the VN (VPath), edge disjoint from $(\hat{u}, \hat{v})$ such that $b_{\hat{u}\hat{v}}$ bandwidth is available between VNodes $\hat{u}$ and $\hat{v}$ even 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.
- Map each VLink $(\hat{u}, \hat{v}) \in \hat{E}$ onto 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 provisioned on $(\hat{u}, \hat{v})$.
- A VLink $(\hat{u}, \hat{v}) \in \hat{E}$ and the VLinks on its backup VPath $\hat{P}_{\hat{u}\hat{v}}$ are edge disjointedly mapped to ensure a single SLink failure does not affect them at the same time.
- The total cost of allocating bandwidth on the SN to embed the VN along with the spare bandwidth is minimum.

C. Shared Risk Group

VLinks that share at least one SLink on their mapped SPaths share the risk of failure since all of them can fail if the shared SLink fails. In a context where only SLink failure is considered, a set of VLinks belong to the same shared risk group (SRG) *iff* they share at least one SLink on their mapped SPaths. On the other hand, VLinks that do not share any SLink on their mapped SPaths belong to different SRGs. To represent the SRG memberships, 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 decision variable to decide on a VLink's membership to an SRG:

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

D. Spare Capacity Assignment Model

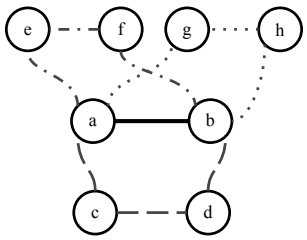
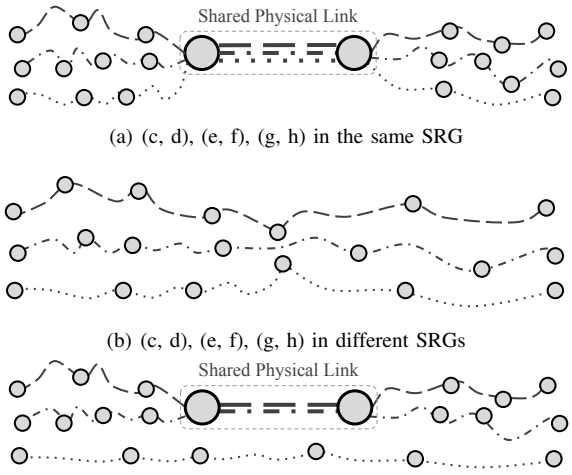


Fig. 1. VLink (a, b) on backup VPaths of VLinks (c, d), (e, f), (g, h)

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. 1. In this figure, VLink (a, b) is on the backup VPaths of three other VLinks: (c, d), (e, f), and (g, h). 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:

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. 2(a)). A single substrate 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.*, $b_{cd} + b_{ef} + b_{gh}$.



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

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

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

Fig. 2. Different Physical Embedding of the VLinks in Fig. 1

All three belong to different SRG. If all three VLinks belong to different SRGs, then they do not share any SLink on their mapped SPaths (Fig. 2(b)). At most one of the VLinks will be affected by a single SLink failure. Therefore, the spare backup capacity allocated on (a, b) should be sufficient to support the maximum bandwidth requirement of these three VLinks, *i.e.*, $\max(b_{cd}, b_{ef}, b_{gh})$.

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. 2(c), 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, spare capacity allocated on (a, b) 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 $D = \{d_1, d_2, d_3, \dots d_{|D|}\}$ SRGs, we can generalize the spare backup bandwidth allocated to $(\hat{u}, \hat{v})$ as:

$$S_{\hat{u}\hat{v}} = \max_{\forall d_i \in D} \left(\sum_{\forall (\hat{x},\hat{y}) \in \hat{\mathcal{H}}_{\hat{u}\hat{v}}} d_i^{\hat{x}\hat{y}} b_{\hat{x}\hat{y}} \right) \tag{1}$$

IV. Problem Formulation

We first provide a Quadratic Integer Program (QIP) formulation for the joint spare capacity allocation and survivable embedding problem in § IV-A, followed by a discussion on the complexity of the QIP in § IV-B. We then describe the transformation of the QIP to an ILP in § IV-C.

A. Quadratic Integer Program Formulation

We first present our decision variables (§ IV-A1). Then we introduce the constraints (§ IV-A2) followed by the objective function of our formulation ((§ IV-A3).

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 that VLink from a single SLink failure. When any SLink on the VLink's mapped SPath fails, $\hat{P}_{\hat{u}\hat{v}}$ provides the same 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:

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

VLinks that share at least one SLink on their mapped SPaths belong to the same SRG (§ III). SRG membership is defined using the decision variable $d_i^{\hat{u}\hat{v}}$, defined in § III-C.

2) Constraints:

a) VNode Mapping Constraints: (2) and (3) ensure that each VNode of a VN is provisioned on an SNode satisfying the provided location constraints. Moreover, (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.

$$\forall \hat{u} \in \hat{V}, \forall u \in V : y_{\hat{u}u} \leq \ell_{\hat{u}u} \tag{2}$$

$$\forall \hat{u} \in \hat{V} : \sum_{u \in V} y_{\hat{u}u} = 1 \tag{3}$$

$$\forall u \in V : \sum_{\hat{u} \in \hat{V}} y_{\hat{u}u} \leq 1 \tag{4}$$

b) Backup VPath Continuity Constraints: A VLink in a VN is protected by a VPath in the VN to survive a single SLink failure. (5) ensures continuity of a backup VPath protecting a VLink $(\hat{x}, \hat{y}) \in \hat{E}$:

$$\forall (\hat{x}, \hat{y}) \in \hat{E} : \sum_{\forall \hat{v} \in \mathcal{N}(\hat{u}) \setminus \{\hat{y}\}} (z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}} - z_{\hat{x}\hat{y}}^{\hat{v}\hat{u}}) = \begin{cases} 1 & \text{if } \hat{u} = \hat{x} \\ -1 & \text{if } \hat{u} = \hat{y} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

c) VLink Mapping Constraints: First, we ensure that every VLink is mapped to a non-empty set of SLinks using (6). Then, we ensure that the in-flow and out-flow of each SNode is equal, except for the SNodes where the endpoints of a VLink are mapped using (7). (7) ensures that the non-empty set of SLinks corresponding to a VLink's mapping form a single continuous SPath.

$$\forall (\hat{u}, \hat{v}) \in \hat{E} : \sum_{\forall (u,v) \in E} x_{uv}^{\hat{u}\hat{v}} \geq 1 \tag{6}$$

$$\forall \hat{u}, \hat{v} \in \hat{V}, \forall u \in V : \sum_{\forall v \in \mathcal{N}(u)} (x_{uv}^{\hat{u}\hat{v}} - x_{vu}^{\hat{u}\hat{v}}) = y_{\hat{u}u} - y_{\hat{v}u} \tag{7}$$

The binary nature of the VLink mapping decision variable and the flow constraint prevents any VLink from being mapped onto more than one SPaths, thus, restricting the VLink mapping to the *Multi-commodity Unsplittable Flow Problem* [26].

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 (1) as follows:

$$S_{\hat{u}\hat{v}} = \max_{\forall d_i} \sum_{\forall (\hat{x},\hat{y}) \in \hat{E} \setminus \{(\hat{u},\hat{v})\}} z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}} \times d_i^{\hat{x}\hat{y}} \times b_{\hat{x}\hat{y}} \tag{8}$$

Then, the following constraint prevents any over-commit of the bandwidth resource in the SLinks:

$$\forall (u,v) \in E : \sum_{\forall (\hat{u},\hat{v}) \in \hat{E}} x_{uv}^{\hat{u}\hat{v}} \times (b_{\hat{u}\hat{v}} + S_{\hat{u}\hat{v}}) \leq b_{uv} \tag{9}$$

Note that (9) is a cubic constraint, since $S_{\hat{u}\hat{v}}$ is quadratic according to (8). Therefore, we take the following steps to linearize $S_{\hat{u}\hat{v}}$ in order to ensure that (9) remains quadratic.

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 such that they 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:

$$\forall (\hat{u},\hat{v}) \in \hat{E}, \forall (\hat{x},\hat{y}) \in \hat{E}, \forall d_i \in D : z_{\hat{x}\hat{y}}^{\hat{u}\hat{v}} + d_i^{\hat{x}\hat{y}} + g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}} \leq 2 \tag{10}$$

We can use $g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i)$ to rewrite (8) in a linear form as follows:

$$S_{\hat{u}\hat{v}} = \max_{\forall d_i} \sum_{\forall (\hat{x},\hat{y}) \in \hat{E} \setminus \{(\hat{u},\hat{v})\}} (1 - g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i)) \times b_{\hat{x}\hat{y}} \tag{11}$$

Since our objective function will be a minimization function, we define $g_{\hat{x}\hat{y}}^{\hat{u}\hat{v}}(i)$ so that setting it to 1 minimizes the value of $S_{\hat{u}\hat{v}}$, unless it is constrained to be 0 according to (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 defined by (10).

d) Disjointedness 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 ensured by (12). (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 we ensure by (14).

$$\begin{aligned} &\forall (u,v) \in E, \forall (\hat{u},\hat{v}) \in d_i, \forall (\hat{x},\hat{y}) \in d_j \textit{ s.t. } i \neq j : \\ &d_i^{\hat{u}\hat{v}} + d_j^{\hat{x}\hat{y}} + x_{uv}^{\hat{u}\hat{v}} + x_{vu}^{\hat{u}\hat{v}} + x_{uv}^{\hat{x}\hat{y}} + x_{vu}^{\hat{x}\hat{y}} \leq 3 \end{aligned} \tag{12}$$

$$\begin{aligned} &\forall d_i \in D, \forall ((\hat{u},\hat{v}),(\hat{x},\hat{y})) \in d_i \times d_i \textit{ s.t. } (\hat{u},\hat{v}) \neq (\hat{x},\hat{y}), \\ &\exists (u,v) \in E : d_i^{\hat{u}\hat{v}} + d_i^{\hat{x}\hat{y}} + x_{uv}^{\hat{u}\hat{v}} + x_{vu}^{\hat{u}\hat{v}} + x_{uv}^{\hat{x}\hat{y}} + x_{vu}^{\hat{x}\hat{y}} = 4 \end{aligned} \tag{13}$$

$$\forall (\hat{x},\hat{y}) \in \hat{E} : \sum_{\forall d_i \in D} d_i^{\hat{x}\hat{y}} = 1 \tag{14}$$

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 constraint ensures this disjointedness:

$$\begin{aligned} &\forall (u,v) \in E, \forall ((\hat{u},\hat{v}),(\hat{x},\hat{y})) \in \hat{E} \times \hat{E} \textit{ s.t. } (\hat{u},\hat{v}) \neq (\hat{x},\hat{y}) : \\ &z_{\hat{u}\hat{v}}^{\hat{x}\hat{y}} + x_{uv}^{\hat{u}\hat{v}} + x_{vu}^{\hat{u}\hat{v}} + x_{uv}^{\hat{x}\hat{y}} + x_{vu}^{\hat{x}\hat{y}} \leq 2 \end{aligned} \tag{15}$$

3) Objective Function: As per the problem statement presented in § III-B, 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 backup bandwidth for the VLinks of a VN on the SLinks of an SN. This gives us the following objective function:

$$\text{minimize} \left(\sum_{\forall (\hat{u},\hat{v}) \in \hat{E}} \sum_{\forall (u,v) \in E} x_{uv}^{\hat{u}\hat{v}} \times C_{uv} \times (b_{\hat{u}\hat{v}} + S_{\hat{u}\hat{v}}) \right) \tag{16}$$

B. Complexity of the QIP

Our formulation for the joint optimization problem has a quadratic constraint (9) and a quadratic objective function (16). Therefore, the QIP presented in § IV-A is a Quadratically Constrained Quadratic Program (QCQP) and falls into the general category of the Quadratic Assignment Problem (QAP) [27]. Solving a QAP is computationally expensive and is known to be $\mathcal{NP}$-hard [28]. Sahni *et al.*, proved that even finding an $\epsilon-$approximate solution of QAP is NP-hard [29]. In the next section, we present the steps to linearize the QIP by using a technique similar to the one discussed in [30].

C. ILP Transformation

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 integer. 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 relationship between $q_{uv}^{\hat{u}\hat{v}}$ and $x_{uv}^{\hat{u}\hat{v}}$:

$$\forall (u,v) \in E, \forall (\hat{u},\hat{v}) \in \hat{E} : q_{uv}^{\hat{u}\hat{v}} \geq 0 \tag{17}$$

$$\forall (u,v) \in E, \forall (\hat{u},\hat{v}) \in \hat{E} : S_{\hat{u}\hat{v}} - \lambda \times (1 - x_{uv}^{\hat{u}\hat{v}}) \leq q_{uv}^{\hat{u}\hat{v}} \tag{18}$$

To elaborate, when $x_{uv}^{\hat{u}\hat{v}} = 0$, constraints (17) and (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 number 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 (17) and (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 later case, constraint (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 $q_{uv}^{\hat{u}\hat{v}} = S_{\hat{u}\hat{v}}$.

We now rewrite the capacity constraint (9) as the following linear constraint using $q_{uv}^{\hat{u}\hat{v}}$.

$$\forall (u,v) \in E : \sum_{\forall (\hat{u},\hat{v}) \in \hat{E}} (x_{uv}^{\hat{u}\hat{v}} \times b_{\hat{u}\hat{v}} + q_{uv}^{\hat{u}\hat{v}}) \leq b_{uv} \tag{19}$$

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

$$\text{minimize} \left(\sum_{\forall (\hat{u},\hat{v}) \in \hat{E}} \sum_{\forall (u,v) \in E} x_{uv}^{\hat{u}\hat{v}} \times C_{uv} \times b_{\hat{u}\hat{v}} + C_{uv} \times q_{uv}^{\hat{u}\hat{v}}) \right) \tag{20}$$

V. Heuristic Design

There are several challenges in designing a heuristic for the joint optimization of spare backup capacity allocation and survivable embedding problem. We first discuss these challenges in detail and briefly explain how we addressed them in § V-A. Then, we present our heuristic algorithm in § V-B.

A. Challenges

1) Selection of Backup VPath: The first challenge is the selection of a backup VPath from an exponential number of VPaths. One of the most efficient shared backup path protection schemes for single layer protection is p-cycle based protection [31]. A p-cycle is formed by connecting the spare capacity in a network in a ring like structure. For instance, a Hamiltonian p-cycle that passes through all the nodes in a network once, can provide recovery paths for either an on-cycle VLink failure or a straddling VLink failure [32]. However, finding a Hamiltonian cycle is $\mathcal{NP}$-Complete. Additionally, a longer p-cycle requires all the VLinks on the p-cycle to be mapped on disjoint SPaths to ensure that the VLinks belong to separate SRGs. Such constraints can lead to longer mapped SPaths or even failure to embed the VN. We address this issue by keeping an overlap between the backup allocations and mapping phases. We first find a mapping using estimated backup allocations and then re-optimize the backup allocations using a p-cycle based technique.

2) VNode Mapping: The next challenge comes from node mapping, a combinatorial optimization problem [33]. Although we are not considering any cost for VNode mapping, the order of VNode mapping and VNode mapping itself have a profound impact on subsequent VLink mapping. Selection of an SNode for mapping a VNode influences later VNodes' mappings and the possible SPaths that can be chosen for the VLinks. Such constraints subsequently impact the cost of a solution. In our approach, we map the VNodes from the most constrained to the least constrained, to minimize chances of mapping failure at a later stage due to resource exhaustion. We measure the constraint of a VNode $\hat{u}$ by its nodal degree *i.e.*, $|\mathcal{N}(\hat{u})|$. To find the mapping of a VNode, we select the SNode incurring the least cost from the location constraint set of the VNode to minimize total cost of the embedding.

3) Disjoint SPath Computation: Finally, VLink mapping on unsplittable SPath without the disjointedness constraints is at least as hard as solving the $\mathcal{NP}$-Hard *Multi-commodity Unsplittable Flow* Problem [26]. Furthermore, finding the optimal set of disjoint SPaths is an $\mathcal{NP}$-Complete problem [34]. In our solution, we iteratively compute the disjoint SPaths using a modified version of *Dijkstra's shortest path* algorithm [35].

B. Heuristic Algorithm

Our heuristic algorithm is presented as a pseudocode in Alg. 1. Alg. 1 solves the joint optimization problem in two steps: (i) it estimates the spare backup bandwidth on the VLinks, determines the disjointedness requirements based on this estimation, and performs a VN Embedding, (ii) it identifies the longest cycle consisting of VLinks belonging to separate SRGs and re-optimizes backup bandwidth allocations using the cycle. Upon success, Alg. 1 returns $nmap$, $emap$, $backup$, and S representing the VNode mapping, VLink mapping, backup VPaths, and spare backup capacities, respectively.

Algorithm 1: Embed VN with Protection

function *VNEmbedding(G, $\hat{G}$, C, σ)*
  $D \leftarrow \{d_1, d_2, \dots d_{|\hat{E}|}\}$ `// Set of all SRGs`
  $\forall \hat{u} \in \hat{V}$: $nmap_{\hat{u}} \leftarrow$ NIL
  $\forall (\hat{u}, \hat{v}) \in \hat{E}$: $S^{est}_{\hat{u}\hat{v}} \leftarrow 0$, $\Lambda_{\hat{u}\hat{v}} \leftarrow \sigma$, $SRG_{\hat{u}\hat{v}} \leftarrow d_1$, $emap_{\hat{u}\hat{v}} \leftarrow \phi$, $backup_{\hat{u}\hat{v}} \leftarrow \phi$
  $\hat{\mathcal{V}} \leftarrow$ Sort $\hat{u} \in \hat{V}$ in decreasing order of $|\mathcal{N}(\hat{u})|$
  foreach $\hat{u} \in \hat{\mathcal{V}}$ **do**
    foreach $\hat{v} \in \mathcal{N}(\hat{u})$ **do**
      $backup_{\hat{u}\hat{v}} \leftarrow$ GetBackup($\hat{G}, (\hat{u}, \hat{v}), \Lambda, emap$)
      foreach $(\hat{x}, \hat{y}) \in backup_{\hat{u}\hat{v}}$ **do**
        $S^{est}_{\hat{x}\hat{y}} \leftarrow \max(S^{est}_{\hat{x}\hat{y}}, b_{\hat{u}\hat{v}})$
        if $SRG_{\hat{u}\hat{v}} = SRG_{\hat{x}\hat{y}}$ **then**
          Find $d_j \in D$ *s.t.* $SRG_{\hat{u}\hat{v}} \neq d_j, \forall (\hat{u}, \hat{v}) \in \hat{E}$
          $SRG_{\hat{u}\hat{v}} \leftarrow d_j$
        $\hat{\mathcal{H}}_{\hat{x}\hat{y}} \leftarrow \{(\hat{a}, \hat{b}) \in \hat{E} | (\hat{x}, \hat{y}) \in backup_{\hat{a}\hat{b}}\}$
        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.* $SRG_{\hat{u}\hat{v}} \neq d_j, \forall (\hat{u}, \hat{v}) \in \hat{E}$
            $SRG_{\hat{u}\hat{v}} \leftarrow d_j$
    $best_{\hat{u}} \leftarrow$ 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**
        $W \leftarrow C$
        $\forall (m, n) \in \{(u, v) \in E | SRG_{\hat{x}\hat{y}} \neq SRG_{\hat{u}\hat{v}} \wedge (u, v) \in emap_{\hat{x}\hat{y}}, \forall (\hat{x}, \hat{y}) \in \hat{E}\}$: $W_{mn} \leftarrow \infty$
        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)
  return $\{nmap, emap, backup, S\}$

Alg. 1 starts by initializing the estimated spare backup bandwidth of each VLink, $S^{est}_{\hat{u}\hat{v}}$ to 0 and by placing all the VLinks into a single SRG d_1. It then proceeds to map the VNodes from the most constrained to the least constrained ones, *i.e.*, in decreasing order of their degrees. If two VNodes have equal degrees then we arbitrarily select one of them. For a VNode $\hat{u}$, Alg. 1 first finds estimated backup VPaths for each VLink incident to $\hat{u}$ by iteratively invoking GetBackup procedure (Alg. 2). Alg. 2 invokes Constrained Weighted Shortest Path (CWSP) procedure to compute a VPath with at

Algorithm 2: Compute Backup VPath of a VLink

1 **function** *GetBackup*($\hat{G}, (\hat{u}, \hat{v}), \Sigma, emap$)
2 **foreach** $(\hat{x}, \hat{y}) \in \hat{E}$ **do**
3 **if** $S^{est}_{\hat{x}\hat{y}} \geq b_{\hat{u}\hat{v}}$ **then** $Weight^{est}_{\hat{x}\hat{y}} \leftarrow 1$
4 **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** $Weight^{est}_{\hat{x}\hat{y}} \leftarrow (b_{\hat{u}\hat{v}} - S^{est}_{\hat{x}\hat{y}}) \times \Sigma_{\hat{x}\hat{y}}$
5
6 **else** $Weight^{est}_{\hat{x}\hat{y}} \leftarrow \infty$
7 $Weight^{est}_{\hat{u}\hat{v}} \leftarrow \infty$
8 **return** CWSP($\hat{G}, \hat{u}, \hat{v}, b_{\hat{u}\hat{v}}, Weight^{est}$)

Algorithm 3: Reconfigure Backup VPaths of all VLinks

1 **function** *UpdateBackup*($\hat{G}, backup, SRG$)
2 $\forall(\hat{u}, \hat{v}) \in \hat{E} : S_{\hat{u}\hat{v}} \leftarrow 0$
3 $\hat{R} \leftarrow$ longest cycle in $\hat{G}$ such that no VLink pair in $\hat{R}$ shares an SLink on their mapped SPaths
4 $\forall(\hat{x}, \hat{y}) \in \hat{R} : S_{\hat{x}\hat{y}} \leftarrow \max_{\forall(\hat{u},\hat{v})\in\hat{R}}\{b_{\hat{u}\hat{v}}\}$
5 **foreach** $(\hat{u}, \hat{v}) \in \hat{E}$ **do**
6 **foreach** $(\hat{x}, \hat{y}) \in \hat{E} \setminus \{(\hat{u}, \hat{v})\}$ **do**
7 **if** $SRG_{\hat{u}\hat{v}} = SRG_{\hat{x}\hat{y}}$ **then** $Weight_{\hat{x}\hat{y}} \leftarrow \infty$
8 **else if** $S_{\hat{x}\hat{y}} \geq b_{\hat{u}\hat{v}}$ **then** $Weight_{\hat{x}\hat{y}} \leftarrow 1$
9 **else** $Weight_{\hat{x}\hat{y}} \leftarrow (b_{\hat{u}\hat{v}} - S_{\hat{x}\hat{y}})$
10 $Weight_{\hat{u}\hat{v}} \leftarrow \infty$
11 $backup_{\hat{u}\hat{v}} \leftarrow$ CWSP($\hat{G}, \hat{u}, \hat{v}, b_{\hat{u}\hat{v}}, Weight$)
12 $\forall(\hat{x}, \hat{y}) \in backup_{\hat{u}\hat{v}} : S_{\hat{x}\hat{y}} \leftarrow \max(S_{\hat{x}\hat{y}}, b_{\hat{u}\hat{v}})$
13 **return** $\{backup, S\}$

least $b_{\hat{u}\hat{v}}$ bandwidth between $\hat{u}$ and $\hat{v}$ in the VN $\hat{G}$, according to a provided weight function. Alg. 2 first computes the weight function $Weight^{est}$ for all the VLinks and invokes CWSP to obtain the backup VPath between $\hat{u}$ and $\hat{v}$. Alg. 2 assigns lower weights to VLinks with already assigned backups to enhance the sharing of spare bandwidth by more VPaths (Line 3). The weight function also takes the mapping cost of an already mapped VLink $(\hat{x}, \hat{y})$ into account and assigns $(\hat{x}, \hat{y})$ a weight proportional to the mapping cost $\Lambda_{\hat{x}\hat{y}}$. In line 4, a special case occurs when a VLink $(\hat{x}, \hat{y})$ is not yet mapped. For this case, we set $\Lambda_{\hat{x}\hat{y}}$ to use the average SPath length (σ) as an indicator of future cost. 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 6).

After computing the estimated backup VPath $backup^{est}_{\hat{u}\hat{v}}$, Alg. 1 updates $S^{est}_{\hat{x}\hat{y}}$ for all $(\hat{x}, \hat{y}) \in backup^{est}_{\hat{u}\hat{v}}$ with the maximum value of $b_{\hat{u}\hat{v}}$ (Line 10). It then places $(\hat{u}, \hat{v})$ and $(\hat{x}, \hat{y})$ into different SRGs (Line 13). Finally, it places $(\hat{u}, \hat{v})$ and all other VLinks that use $(\hat{x}, \hat{y})$ in their backup VPaths into different SRGs (Line 17). After finding the backup VPaths and SRGs of all the incident VLinks of a VNode $\hat{u}$, Alg. 1 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 $20-29$). 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. 1 computes CWSP from l to $nmap_{\hat{v}}$ (Line 24), while satisfying capacity constraints and SRG constraints in the SN (using the weights in W). To do so, Alg. 1 identifies the set of SLinks that the mapping of $(\hat{u}, \hat{v})$ should be disjoint from and assigns ∞ as their weights (Line 22). 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 25). After mapping a VNode $\hat{u}$, Alg. 1 maps the VLinks whose both endpoints have already been mapped and updates Λ of the mapped VLinks (Line 33).

The last phase (Alg. 3) of our heuristic leverages the concept of p-cycle based protection to optimize the spare backup bandwidth $S_{\hat{u}\hat{v}}$ for each mapped VLink $(\hat{u}, \hat{v}) \in \hat{E}$. Alg. 3 first finds the longest cycle $\hat{R}$ in $\hat{G}$ such that any pair of VLinks in $\hat{R}$ do not share any SLink in their mappings (Line 3). Recall from § III-D that each $(\hat{x}, \hat{y}) \in \hat{R}$ belongs to distinct SRGs for a single SLink failure. Therefore, Alg. 3 allocates the maximum of the demands of all the VLinks in $\hat{R}$ to each $S_{\hat{x}\hat{y}} \in \hat{R}$ (Line 4). 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. 2. However, Alg. 3 utilizes the mapping 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 7). Alg. 3 also enhances the spare capacity sharing by setting unit weights to the VLinks having already assigned spare capacities (Line 8). Alg. 3 then invokes the CWSP procedure with the weight function to compute $backup_{\hat{u}\hat{v}}$. Finally, $S_{\hat{x}\hat{y}}$ for each $(\hat{x}, \hat{y}) \in backup_{\hat{u}\hat{v}}$ is updated accordingly (Line 12).

C. Running Time Analysis

The CWSP procedure is implemented using a modified *Dijkstra's shortest path* algorithm, taking into account the constraints and weights. *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. 1, 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 $O(|\hat{V}|\mathcal{L}\delta^2(|E| + |V| \log |V|))$.

VI. Evaluation

We evaluate our proposed solutions for the joint optimization problem through extensive simulations. We briefly discuss the compared approaches in § VI-A, simulation setup in § VI-B followed by the performance metrics in § VI-C. Finally, we describe our evaluation results focusing on the following aspects: (i) impact of SN (§ VI-D), (ii) impact of VN (§ VI-E), and (ii) scalability of the solutions (§ VI-F).

A. Compared Approaches

We implemented the ILP-based optimal solution, *Opt-ILP*, presented in IV-C using IBM ILOG CPLEX C++ libraries and compare that with a C++ implementation of the *heuristic*. However, *Opt-ILP* was unable to scale beyond very small problem instances. Therefore, we implemented a simpler variant of *Opt-ILP*, called *Max-ILP*, to use as a baseline. Design of *Max-ILP* is motivated by an observation from the results that for a VLink $(\hat{u}, \hat{v})$, *Opt-ILP* places the VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$ into separate SRGs whenever possible, thus preferring more

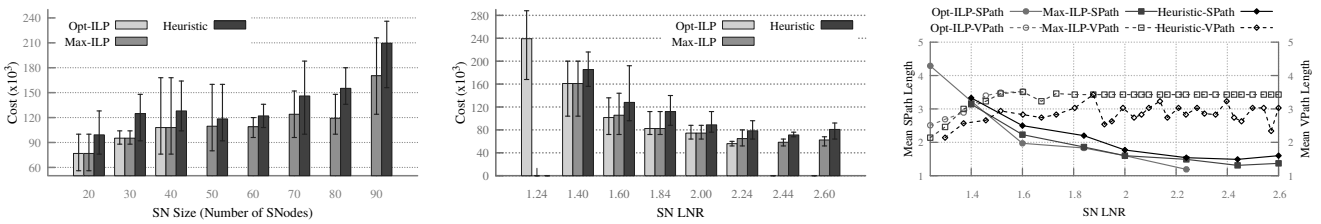


Fig. 3. Impact of SN Topology

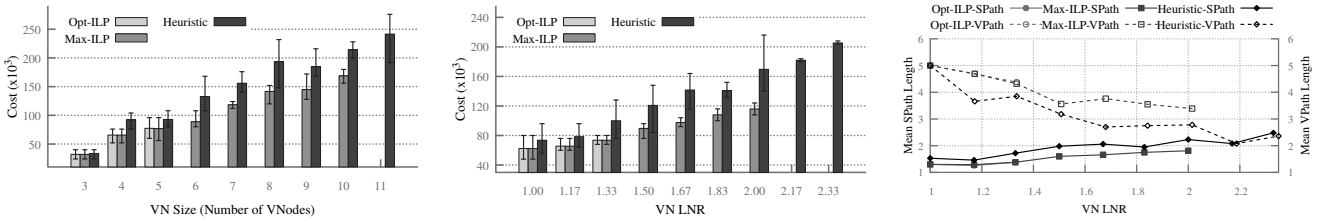


Fig. 4. Impact of VN Topology

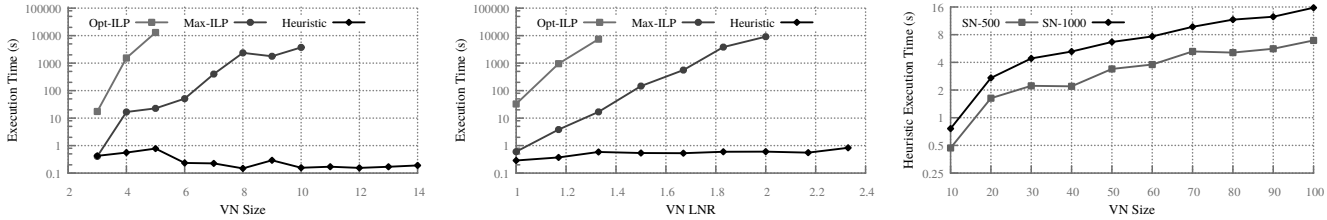


Fig. 5. Scalability Analysis

sharing of the spare capacity. Hence, we constrained any pair of VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$ to be in separate SRGs during embedding. Doing so allowed us to exclude the decision variable $d_i^{\hat{x}\hat{y}}$ from (8) and reduce complexity. For *Max-ILP*, (8) was modified to take the maximum demand of the VLinks in $\hat{\mathcal{H}}_{\hat{u}\hat{v}}$ as $S_{\hat{u}\hat{v}}$.

B. Simulation Setup

We evaluate the compared approaches on both small and large scale settings. Since VNs are still not widely deployed, the topological properties of VNs and SNs are not well understood yet. Hence, we vary number of nodes (size) and link to node ratio (LNR) of VNs and SNs. For each simulation run, we generate an SN and 5 random VNs with the desired property. In small scale, SN size is varied between 20 – 90 nodes, while VN size ranges from 3 – 11 nodes. For larger scale, VN size ranges from 10 – 100 nodes on 500 and 1000 node SNs. We vary the connectivity of both SNs and VNs by varying the LNR from 1.0 to 2.60. We set VLink demand 10% of the SLink bandwidth. For each SN, the performance metrics are measured by taking the mean over all 5 VNs. Simulations are performed on a machine with $2\times$8-core 2.0 Ghz Intel Xeon E5-2650 processors and 256GB of RAM.

C. Performance Metrics

1) Cost: The cost of embedding a VN computed using (20). We set a unit cost for allocating bandwidth on an SLink, therefore, (20) directly represents resource consumption.

2) Execution Time: The time required for an algorithm to find an embedding of a VN.

3) Mean SPath Length: The mean length of the SPath used to map a VLink of a VN.

4) Mean VPath Length: The mean length of the VPath used as a backup path for a VLink in a VN.

D. Impact of SN Topology

Fig. 3(a) presents embedding costs for different SN sizes, while keeping the VN size and SN LNR fixed at 5 and 1.8, respectively. We first observe that *Max-ILP* very closely approximates *Opt-ILP*. Compared to *Max-ILP*, the *heuristic* provisions $\sim 21\%$ additional resources on average over all test cases. For a fixed LNR, with increasing SN size, embedding costs increase for all approaches. As SN sizes increase, candidate SNodes of the VNodes of a VN are placed far apart from one another, thus, contributing to larger costs. However, for a fixed SN size, with increasing SN LNR, costs decrease for all three approaches as observed in Fig. 3(b). We explain this behavior with the help of Fig. 3(c) in the following.

Fig. 3(c) presents mean SPath and VPath lengths by varying SN LNR. At the lowest end of the LNR spectrum, *Max-ILP* and *heuristic* fail to find sufficient disjoint SPaths, imposed by the SRG constraints, resulting in infeasible solutions. However, *Opt-ILP* is able to find a solution with a very high cost by reducing the number of SRGs. Reducing the number of SRGs is beneficial in SNs with lower LNRs, since it is hard to find sufficient disjoint SPaths in such SNs. In addition, a disjoint SPath becomes significantly longer than the corresponding non-disjoint SPath between the same pair of SNodes in an SN with lower LNR. As SN LNR increases, the number and length of the disjoint SPaths increase and decrease, respectively. As a consequence, cost decreases for all three approaches as shown in Fig. 3(b). Fig. 3(c) shows that mean VPath lengths for all three approaches increase initially with increasing SN LNR. However, when VPath lengths get very close to the VN diameter, 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 sharing of spare backup bandwidth. On the other hand, SNs with lower LNRs cannot satisfy the disjointedness constraints imposed by longer VPaths, hence, all the approaches select shorter VPaths resulting in more spare bandwidth and more cost (Fig. 3(b)).

E. Impact of VN Topology

Fig. 4(a) presents embedding costs for different VN sizes, while keeping the SN size, SN LNR, and VN LNR fixed at 50, 1.82, and ~ 1.4, respectively. Fig. 4(b) and Fig. 4(c) compare embedding cost, mean SPath and VPath lengths by varying the VN LNR. The key takeaway from these figures is that both cost and mean SPath length increase with increasing VN size and VN LNR. This is due to more disjointedness constraints imposed by the higher number of SRGs and shorter VPaths induced by both larger and denser VNs (Fig. 4(c)).

F. Scalability Analysis

Fig. 5 shows the execution times for the compared approaches to demonstrate their scalability. As the problem size increases in terms of VN size, VN LNR, and SN size, the execution time grows for all the approaches except for the case of *heuristic* execution times against VN size. The decrease in *heuristic*'s execution times with increasing VN size is due to the reduction of the SN solution space imposed by the higher number of SRG constraints. In contrast, the execution time of *Opt-ILP* and *Max-ILP* increase exponentially, limiting their applicability to smaller problem instances. As we can see from Fig. 5(a) and Fig. 5(b), with our current hardware, both *Opt-ILP* and *Max-ILP* hit a ceiling in terms of VN size or VN LNR. Whereas, the *heuristic* can solve much larger problem instances. Even for the successful cases, *Opt-ILP* and *Max-ILP* require several orders of magnitude more time to solve similar problem instances compared to the *heuristic*. Furthermore, the *heuristic* is able to find solutions within a reasonable time limit for much larger problem instances (Fig. 5(c)), *i.e.*, VNs with 10–100 nodes and 21–285 links on 500 and 1000 node SNs.

VII. Conclusion and Future Work

In this paper, we have proposed a novel solution to the SVNE problem. Instead of addressing the problem at the SN level, we have addressed the problem at the VN level, *i.e.*, the VN is augmented with sufficient spare backup bandwidth and embedded on the SN accordingly to ensure survivability against 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 also proposed a heuristic to tackle the computational complexity of the optimal solution. Simulation results show that our heuristic allocates $\sim$21% additional resources compared to the optimal solution, while executing several orders of magnitude faster. In the future, we plan to extend this work to virtual fabrics where the bandwidth requirement is expressed as pairwise bandwidth rather than as per link bandwidth requirement. We intend to study the resource allocation challenges to ensure survivability at the fabric level compared to doing the same at the VN level.

References

[1] "Amazon Virtual Private Cloud," http://aws.amazon.com/vpc/.

[2] "T-SDN Prototype Demonstration," https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/oif-p0105_031_18.pdf.

[3] N. M. M. K. Chowdhury *et al.*, "A Survey of Network Virtualization," *Computer Networks*, Apr 2010.

[4] M. R. Rahman, I. Aib, and R. Boutaba, "Survivable virtual network embedding," in *NETWORKING 2010*. Springer, 2010, pp. 40–52.

[5] J. Xu *et al.*, "Survivable virtual infrastructure mapping in virtualized data centers," in *IEEE CLOUD*. IEEE, 2012, pp. 196–203.

[6] H. Yu *et al.*, "Cost efficient design of survivable virtual infrastructure to recover from facility node failures," in *IEEE ICC*, 2011, pp. 1–6.

[7] N. Shahriar *et al.*, "Connectivity-aware virtual network embedding," in *IFIP Networking Conference 2016*, 2016, pp. 46–54.

[8] M. M. A. Khan *et al.*, "Simple: Survivability in multi-path link embedding," in *IEEE CNSM*, 2015, pp. 210–218.

[9] R. R. Oliveira *et al.*, "Dos-resilient virtual networks through multipath embedding and opportunistic recovery," in *ACM SAC '13*, pp. 597–602.

[10] T. Guo *et al.*, "Shared Backup Network Provision for Virtual Network Embedding," in *IEEE ICC*, Kyoto, Japan, Jun 2011, pp. 1–5.

[11] Y. Chen *et al.*, "Resilient Virtual Network Service Provision in Network Virtualization Environments," in *IEEE ICPADS '10*, pp. 51–58.

[12] Z. Ye *et al.*, "Survivable virtual infrastructure mapping with dedicated protection in transport software-defined networks [invited]," *Journal of Optical Comm. and Net.*, vol. 7, no. 2, pp. A183–A189, 2015.

[13] S. R. Chowdhury *et al.*, "Protecting Virtual Networks with DRONE," in *IEEE/IFIP NOMS*, 2016, pp. 78–86.

[14] I. B. Barla *et al.*, "Optimal design of virtual networks for resilient cloud services," in *9th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE, 2013, pp. 218–225.

[15] W. Wang *et al.*, "First Demonstration of Virtual Transport Network Services With Multi-layer Protection Schemes over Flexi-grid Optical Networks," *IEEE Comm. Letters*, vol. 20, no. 2, pp. 260–263, Feb 2016.

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

[17] T. 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*, 2011, pp. 1–8.

[18] D. D.-J. Kan *et al.*, "Lightpath routing and capacity assignment for survivable ip-over-wdm networks," in *IEEE DRCN*, 2009.

[19] P. Gill *et al.*, "Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications," in *ACM SIGCOMM*, vol. 41, Aug 2011, pp. 350–361.

[20] A. Markopoulou *et al.*, "Characterization of Failures in an IP Backbone," in *INFOCOM*, vol. 4, Mar 2004, pp. 123–133.

[21] P. Demeester *et al.*, "Resilience in multilayer networks," *IEEE Comm. Magazine*, vol. 37, no. 8, pp. 70–76, Aug 1999.

[22] C. Assi *et al.*, "On the merit of ip/mpls protection/restoration in ip over wdm networks," in *IEEE GLOBECOM '01*, pp. 65–69.

[23] E. Kubilinskas and M. Pioro, "Two design problems for the ip/mlps over wdm networks," in *DRCN '05*, pp. 241–248.

[24] Y. Liu *et al.*, "Spare capacity allocation in two-layer networks," *IEEE JSAC*, vol. 25, no. 5, pp. 974–986, 2007.

[25] O. Gerstel *et al.*, "Multi-layer capacity planning for ip-optical networks," *IEEE Communications Magazine*, vol. 52, no. 1, pp. 44–51, 2014.

[26] S. Even *et al.*, "On the complexity of time table and multi-commodity flow problems," in *IEEE FOCS*, 1975, pp. 184–193.

[27] E. L. Lawler, "The quadratic assignment problem," *Management science*, vol. 9, no. 4, pp. 586–599, 1963.

[28] E. Cela, *The quadratic assignment problem: theory and algorithms*. Springer Science & Business Media, 2013, vol. 1.

[29] S. Sahni and T. Gonzalez, "P-complete approximation problems," *Journal of the ACM (JACM)*, vol. 23, no. 3, pp. 555–565, 1976.

[30] M. Oral and O. Kettani, "A linearization procedure for quadratic and cubic mixed-integer problems," *Operations Research*, vol. 40, no. 1-supplement-1, pp. S109–S116, 1992.

[31] R. Asthana *et al.*, "p-cycles: An overview," *IEEE Comm. Surveys & Tutorials*, vol. 12, no. 1, pp. 97–111, 2010.

[32] W. D. Grover and D. Stamatelakis, "Cycle-oriented distributed preconfiguration: ring-like speed with mesh-like capacity for self-planning network restoration," in *IEEE ICC*, 1998, pp. 537–543.

[33] A. Fischer *et al.*, "Virtual Network Embedding: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 15, pp. 1888–1906, 2013.

[34] J. M. Kleinberg, "Approximation algorithms for disjoint paths problems," Ph.D. dissertation, Citeseer, 1996.

[35] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.