Progressive Slice Recovery with Guaranteed Slice Connectivity after Massive Failures

Qiaolun Zhang, Omran Ayoub, Jun Wu, Francesco Musumeci, Gaolei Li and Massimo Tornatore

***Abstract*—In presence of multiple failures affecting their network infrastructure, operators are faced with the Progressive Network Recovery (PNR) problem, i.e., deciding the best sequence of repairs during recovery. With incoming deployments of 5G networks, PNR must evolve to incorporate new recovery opportunities offered by network slicing. In this study, we introduce the new problem of Progressive Slice Recovery (PSR), which is addressed with eight different strategies, i.e., allowing or not to change slice embedding during the recovery, and/or by enforcing different versions of slice connectivity (i.e., network vs. content connectivity). We propose a comprehensive PSR scheme, which can be applied to all recovery strategies and achieves fast recovery of slices. We first prove the PSR's NP-hardness and design an integer linear programming (ILP) model, which can obtain the best recovery sequence and is extensible for all the recovery strategies. Then, to address scalability issues of the ILP model, we devise an efficient two-phases progressive slice recovery (2-phase PSR) meta-heuristic algorithm, small optimality gap, consisting of two main steps: i) determination of recovery sequence, achieved through a linear-programming relaxation that works in polynomial time; and ii) slice-embedding recovery, for which we design an auxiliary-graph-based column generation to re-embed failed slice nodes/links to working substrate elements within a given number of actions. Numerical results compare the different strategies and validate that amount of recovered slices can be improved up to 50% if operators decide to reconfigure only few slice nodes and guarantee content connectivity.**

***Index Terms*—5G networks, virtual network embedding, progressive network recovery, content connectivity, network connectivity.**

I. Introduction

In case of major network disruptions (caused, e.g., by large-scale disasters such as earthquakes, floods, and targeted attacks) in 5G networks, operators must recover their network infrastructure through a sequence of reparation steps. The problem of optimizing this sequence to maximize the amount of services provided during the recovery is commonly referred to as Progressive Network Recovery (PNR). PNR involves multiple stages, where a stage is an amount of time in which the network operator decides the allocation and placement of reparation resources (e.g., field engineers and network equipment). The objective of PNR is to repair network component (typically a link, or a node) such that the network is recovered as early as possible.

As forthcoming 5G networks will support novel forms of network slicing [1], the PNR problem must be evolved to take advantage of new recovery opportunities that network slicing enables. Hence, in this study, we present and investigate the new problem of Progressive Slice Recovery (PSR). With slicing, operators can offer to their customers "slices" of their physical (or "substrate") network resources, typically in the form of Virtual Networks (VNs), that are appropriately embedded (see [2]–[6] for more details on VN embedding and network virtualization) on a subset of the substrate resources to accommodate the requirements of very diverse services (e.g., new services for online gaming or remote manufacturing control). During the recovery, a damaged slice can be more or less rapidly recovered depending on the constraints that the slice has to obey, e.g., depending on the fact that a slice has to maintain a fixed embedding of its substrate nodes and links, or that a slice can dynamically change its embedding [7], [8].

Hence, the operator can decide to perform slice recovery by adopting several strategies, which result in different versions of the PSR problem. In this study, we consider that a slice can be re-embedded under distinct assumptions. In fact, an operator can decide to maintain the same embedding used before the failure (e.g., due to security or isolation reasons), or it might be allowed to reconfigure only the link embedding, or only node embedding, or both. Note that the amount of reconfiguration of embedding is usually limited, as operators prefer to avoid the disruption time introduced by frequent reconfiguration. Moreover, with several 5G services shifting towards a more content-centric communication paradigm, slices can be embedded and recovered with the goal of ensuring only Content Connectivity (CC) within the slice itself, or targeting the more traditional, yet more constraining, Network Connectivity (NC) [9], [10]. NC is a traditional network-survivability metric, which is guaranteed by ensuring reachability among all network nodes, while CC is a more recent concept, and it is defined as the reachability from all network nodes to at least one (among multiple) content replica [10]. On the one hand, as CC does not constrain all nodes to be connected among them, it requires fewer resources than NC. On the other hand, by only ensuring CC, some services might incur degradation. Yet, CC proved to be of great help to provide relief, especially in case of multiple failures [11].

Examples of PSR with NC and CC: Let us consider the example in Fig. 1, which illustrates, using a recovery sequence, how ensuring NC vs. CC of slices affects the progressive recovery of slices differently. In our example, the substrate network is a 7-node network with 11 edges and 2 data centers (DCs). Three network slices, each consisting of 4 edges and 4 nodes (in which 2 are DC nodes, i.e., nodes that connect to a DC) are initially embedded as marked with different colors

This work was supported in part by the National Natural Science Foundation of China under Grant 61972255. *(Corresponding author: Jun Wu.)*

Qiaolun Zhang, Jun Wu and Gaolei Li are with Shanghai Jiao Tong University, China (email: junwuhn@sjtu.edu.cn).

Omran Ayoub, Francesco Musumeci and Massimo Tornatore are with Politecnico di Milano, Italy.

in the substrate network. For instance, virtual link $(b1, c1)$ in slice 1 is embedded onto links (G, F) and (F, E) in the substrate physical network. A disaster zone (highlighted by an oval shape) causes substrate links (B, C), (C, D), (D, E), and (E, F) and substrate nodes D and F to fail, hence disrupting the three slices. We consider 5 stages for the recovery process, at each of the stages from 1 to 4 we assume the network operator can repair one substrate node and one substrate link while maintaining the initial embedding of each of the slices. Stage 0 does not have any resources, but the operator may decide to do re-embedding. For slice 1, we need to recover one virtual node and one virtual link (e.g., link $(c1, d1)$ and node $d1$) to ensure CC. However, we need to recover one virtual node and two virtual links (e.g., link $(c1, d1)$, $(b1, c1)$, and node $d1$) to ensure NC. Note that different recovery sequences can lead to different outcomes in terms of recovered slices. As an example, we consider two different recovery sequences (shown in the figure) and, at the bottom of the figure, we plot the number of recovered slices, with guaranteed NC and CC, respectively. For both recovery sequences, slices with CC are recovered in fewer stages than slices with NC. The plot also shows that sequence 1 can recover all slices quicker than sequence 2 for CC. Note that, if changing the initial embedding is allowed, the network operator would be able to recover all three slices earlier (in stage 1) by recovering the physical node F and re-embedding the virtual nodes of the slices onto nodes A, B, F, and G.

In this study, we develop an extensible slice recovery scheme applicable to all the different PSR problem versions. After proving that all the problem versions are NP-hard, we introduce an integer linear programming (ILP) formulation, which can be used, with slight modifications, to formally represent all the PSR problems. The formulation aims to maximize the accumulated number of recovered slices while enforcing the connectivity of slices and bounding the number of re-embedding actions. Since all the PSR problem versions are computationally intractable, we also devise a solving approach in two phases, namely, the recovery-sequence-determination phase and failed-node/link-re-embedding phase. The first phase obtains the recovery sequence in the substrate network in polynomial time using a linear relaxation of the ILP. The second phase determines efficient reconfiguration action using an auxiliary-graph-based column generation.

Main technical contributions of this study are summarized as follows. To address re-embedding limitations of the PSR problem, a series of on-demand progressive recovery strategies are proposed, which significantly optimize the recovery outcome after massive failures with a customized tradeoff between the cost of disruption time and slice connectivity. To guarantee the provision of content-based service in 5G, we propose to take the number of slices with guaranteed connectivity constraints as one of the optimization objectives to recover slice nodes and links, which is essential to accelerate the recovery of slices. A scalable solving approach based on an auxiliary-graph-based column generation algorithm is proposed for the PSR, which obtains the recovery sequence of the substrate network in polynomial time and determines efficient re-embedding with guaranteed connectivity constraints. To the best of our knowledge, this article is the first to address the PSR problem using content connectivity.

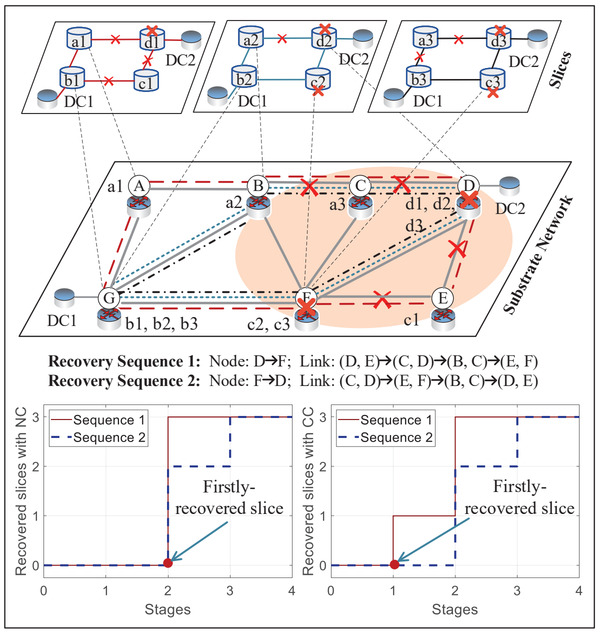


Fig. 1: An example of progressive slice recovery with guaranteed NC and CC.

The rest of the paper is organized as follows. In Section II, we recap existing studies on network resiliency and survivability of network slice. Section III models the PSR problem under reconfiguration limitations. Section IV introduces the generic ILP model applicable to all the problem versions. In Section V, we propose our two-phases scalable solution for the PSR problem. Section VI provides illustrative numerical results. Section VII concludes the paper.

II. Related Work

Most studies on network resiliency and survivability focus on preparing the networks before disaster occurrence, using *proactive protection strategies* [12], [13]. Some studies, as Ref. [14]–[16], focus on over-provisioning resources to protect VNs from failure in the substrate network. Ref. [17] proposed dynamic re-embedding of both virtual nodes and virtual links to re-balance the load on the substrate network in an online setting where new VN requests may arrive or depart. Moreover, various connectivity-guaranteed virtual network embedding (VNE) schemes [9], [10], [18] are proposed, which can be applied before disaster occurrence and are survivable under multiple failures. However, proactive measures cannot guarantee network survivability after massive failures, and hence, *reactive post-failure procedures* must be set in place. Among them, the PNR problem allows to optimally select the sequence of recovery actions during the recovery process [19]–[21]. Even though the PNR has been analyzed in the context of traffic recovery, to the best of our knowledge, no existing research has systematically addressed the PNR problem in the context of slicing/VN recovery with guaranteed connectivity. As an example, Ref. [22] modeled the PNR problem for VNs considering traveling time, but it only covers failure of links and the mapping of nodes is predefined and fixed. Ref. [23] emphasized the importance of the reconfiguration of the VNE in VN recovery, but it did not cover how to sequentially recover the failed substrate elements. Moreover, Ref. [24] considered the role of portable network elements and recovery trucks during slice recovery but it did not address VNE reconfiguration. Similarly, Ref. [25] proposed several heuristics to optimally distribute repair resources, but reconfiguration of the whole VN is allowed at each stage, which is not practical [26].

Different from previous works, we systematically define and classify several strategies for progressive slice recovery. Based on this classification, we designed novel algorithms for all cases and compared, for the first time, the impact of content connectivity with network connectivity during recovery. Another strength of our proposed strategies is that, differently than in previous works, we consider re-embedding and all types of failures, including link, node, and DC failures.

III. Progressive Slice Recovery Problem under Reconfiguration Limitations

A. System Model and Problem Statement

We model the substrate network as a weighted undirected graph $G_p = (N_p, E_p, D_p)$, where N_p is the set of substrate nodes, E_p is the set of substrate links, and D_p is the set of DCs. Every node i is equipped with CPU capacity W_i, and every link (i, j) has capacity $C_{(i,j)}$. Every DC is connected to a substrate node, as shown in Fig. 1. We assume that all the DCs contain all the content required by network slices. To access content from a DC, both the DC and the connected node must be working (not failed). After a large scale failure, we denote the set of failed substrate nodes, links, and DCs as N_f, E_f, and D_f, respectively.

Similarly, we model the network slice as an undirected graph $G_l^v = (N_l^v, E_l^v, D_l^v)$, where N_l^v, E_l^v, and D_l^v represent the set of virtual nodes, virtual links, and DCs in slice v, respectively. Meanwhile, N_l, E_l, D_l denote the set of virtual nodes, virtual links, and DCs of all slices, respectively. V denotes the set of all slices. Every virtual node s can be embedded to one node belonging to the set of candidate substrate nodes $G[s]$, which are geographically close to each other. The capacity requirements of virtual node s and link (s, t) are denoted as w_s and $c_{(s,t)}$, respectively.

Without loss of generality, we can divide the recovery process into K stages. At each stage, a given amount of reparation resources is available. We denote by H_k, R_k, and G_k the amount of resources for substrate nodes, links, and DCs, respectively, at stage $k \in K$. Substrate node $i \in N_p$, link $(i, j) \in E_p$, and DC $d \in D_p$ require h_i, $r_{(i,j)}$, and g_d amount of reparation resources to be repaired. We assume that a substrate component cannot be partially repaired and, if resources are not utilized in a stage, they can be used during following stages. After recovering substrate elements, (re)-embedding may be performed to embed the failing slices to working substrate components. Note that, while the ultimate goal is to recover all the slices, we particularly focus on how the nodes, links, and connectivity of these slices are gradually recovered during the K recovering stages.

Tab. I: Proposed Recovery Strategies

Strategies	(Re)-embedding constraints	Allowed reconfiguration actions
Fixed Initial Embedding (Fix-em)	Maintain both initial node and link embedding	Not Allowed
Link Re-embedding (Link-rem)	Maintain only initial node embedding	Only links
Any Re-embedding (Any-rem)	Allow link and node re-embedding	Any link and node
Any Re-embedding(γ_1, γ_2) (Any-rem(γ_1, γ_2))	Allow link and node re-embedding	Any link; γ_1 normal nodes; γ_2 DC nodes

B. Classification of Proposed Recovery Strategies

Recovery strategies differ based on the (re)-embedding constraints and therefore on the allowed reconfiguration actions (see Table I). The (re)-embedding constraint refers to whether the operator wants to maintain the initial embedding of slice nodes and/or links at each stage. The allowed reconfiguration actions define the type (node or link) and amount of reconfiguration actions that can be performed at each stage compared with the previous stage. During the recovery process, the embedding of slice nodes and links can or cannot be changed depending on the strategy adopted by the operators.

The recovery strategies are summarized in Table I:

- *Fixed Initial Embedding* strategy forces to maintain the initial node and link embedding and so it does not allow any reconfiguration.
- *Link Re-embedding* strategy maintains the initial node embedding but allows virtual link reconfiguration (we assume node reconfiguration is more expensive than link reconfiguration [26] due to service disruption cost, hence we allow complete link reconfiguration, as in [17]).
- *Any Re-embedding* strategy allows unlimited re-embedding and reconfiguration of nodes and links at any stage.
- *Any Re-embedding(γ_1, γ_2)* strategy allows unlimited re-embedding and reconfiguration of links and unlimited re-embedding of nodes but limits the node and DC reconfiguration actions at each stage.

More precisely, regarding *Any Re-embedding(γ_1, γ_2)*, at each stage, at most γ_1 normal nodes (i.e., nodes that do not connect to a DC) can be reconfigured, and at most γ_2 DC nodes (i.e., nodes that connect to a DC) can be reconfigured. This additional constraint comes from the observation that a DC node, when being reconfigured, has higher disruption time than a "normal" node, since the routing of all connections to that DC must be reconfigured. Hence, we enforced a different maximum number of reconfigurations for normal nodes (γ_1) and for DC nodes (γ_2) at each stage. We consider also safe to assume that: (*i*) no re-embedding is less expensive (in terms of both recovery cost) than re-embedding, (*ii*) link re-embedding is less expensive than node re-embedding, and (*iii*) normal node re-embedding is less expensive than DC node re-embedding. In conclusion, the four proposed re-embedding strategies will offer different options to the trade-off between recovery cost and recovery time. Note also that the 4 proposed strategies can either enforce NC or CC, so the total number of strategies is 8.

C. NP-hard of Progressive Slice Recovery

Traditional PNR problem to recover the maximum flow in the substrate network was proven as NP-hard in [19] by reducing it to the Set Covering Problem. The PSR problem is different, as it focuses on maximizing recovered slices and involves, in some variants, virtual network re-embedding. We prove in the following that PSR is also NP-hard, even if we fix the link and node embedding by reducing the Weighted Set Covering (WSC) Problem to it. Moreover, if we allow re-embedding, the PSR problem contains the computationally intractable VNE problem. We now show all slice recovery strategies proposed in Table I are NP-hard.

Theorem 1: The PSR problem of maximizing total progressively recovered slices is NP-hard, even for the case of one stage and only link failures.

Proof: Since there is only one stage, the problem is reduced to the VNE problem, which is NP-hard even with a fixed node embedding when the flow is unsplittable [27]. Consequently, *Link Re-embedding*, *Any Re-embedding*, and *Any Re-embedding(γ_1, γ_2)* are NP-hard. Then, we can prove that also *Fixed Initial Embedding* is NP-hard by showing that WSC, a known NP-Complete problem can be reduced to the decision version of the PNR problem with slicing for *Fixed Initial Embedding* with one stage.

The WSC problem is described as follows: Given a set $S = \{s_1, ..., s_p\}$, a set of subsets of $S_1, ..., S_m \subset S$ with weights $w_1, ..., w_m$, and an integer W, does there exist a set $I \in \{S_1, ..., S_m\}$ such that all the elements in S are covered by I and $\sum_{S_i \in I} w_i \leq W$.

Given an instance of WSC, we can construct an instance of the decision version of the PSR problem with only one stage and only link failures as follows. We use R to denote the available resource for links. Each subset of the failed links E_f corresponds to a set S_j, whose weight w_j refers to the sum of the resources needed for all the links in the set to be repaired. The element s_i corresponds to a recovered slice. We denote a set S_j covers an element s_i if recovering the links in the set can recover the slice. The corresponding question becomes whether there exists a recovery solution such that the number of recovered slices is greater or equal to p.

Suppose that the WSC problem has a feasible solution, which includes subsets $S_1, ..., S_{m'}$. In the corresponding PSR problem, this feasible solution implies that there is a solution by repairing all links in the set $\{(i,j) | (i,j) \in S_1 \cup ... \cup S_{m'}\}$. Such a recovery solution will lead to p recovered slices. On the other hand, suppose that the PSR problem has a recovery solution with recovered slices larger or equal than p by using no more than R resources. In the corresponding WSC problem, this implies that there exists a set $I \in \{S_1, ..., S_{m'}\}$ satisfies that all the elements in S are covered by I and $\sum_{S_i \in I} w_i \leq W$. In conclusion, the PSR problem of maximizing total progressively recovered slices is NP-hard, which is applicable to both NC and CC.

IV. ILP Formulations for Progressive Slice Recovery

In this section, we first introduce the variables and constraints of the ILP formulation, and then we describe the additional constraints required by each specific scheme separately.

A. Decision Variables and Recovery Objectives

Decision variables are listed in Table II. If the slice v satisfies the NC or CC required by the operator at stage k, S_v^k equals to 1. Objective function is to maximize accumulated weighted number of recovered slices. Considering that different slices may have different importance/weight (e.g., a slice for entertainment is less important than a slice for emergency communication), we denote the weight for slice v at stage k as $\alpha_{1,v}^k$. Besides, to break ties of different embedding with the same number of working slices, the number of working virtual links are also included in the objective function, giving priority to solutions where a larger number of virtual links are recovered (even if a slice is not completely recovered). The set of unidirectional links in all slices is denoted as $\overline{E}_l$ and the weight of virtual link $(s,t) \in \overline{E}_l$ is denoted as $\alpha_{2,(s,t)}^k$. Parameters are listed in Table III.

Tab. II: Variables Description for the ILP Model

Variable	Description
$m_{i,s}^k$	Binary, equals to 1 if virtual node $s \in N_l$ is embedded to physical node $i \in N_p$ at stage $k \in K$
$q_{(i,j)}^{(s,t),k}$	Binary, equals to 1 if virtual link $(s,t) \in E_l$ is embedded to physical link $(i,j) \in E_p$ at stage $k \in K$
S_v^k	Binary, equals to 1 if slice $v \in V$ satisfies slice connectivity at stage $k \in K$
$\beta_{(i,j)}^k$	Binary, equals to 1 if physical link $(i,j) \in E_p$ is working at stage $k \in K$
$\overline{\beta}_{(s,t)}^k$	Binary, equals to 1 if virtual link $(s,t) \in E_l$ is working at stage $k \in K$
$Q_{d,n}^k$	Binary, equals to 1 if DC $d \in D_l$ is connected to virtual node $n \in N_l$ at stage $k \in K$
$\overline{Q}_n^k$	Binary, equals to 1 if the virtual node n can reach at least one DC at stage $k \in K$
$p_{(i,j)}^k$	Binary, equals to 1 if physical link $(i,j) \in E_p$ is repaired at stage $k \in K$
z_i^k	Binary, equals 1 if physical node $i \in N_p$ is repaired at stage $k \in K$
b_d^k	Binary, equals to 1 if physical DC $d \in D_p$ is repaired at stage $k \in K$

Tab. III: Parameters Description for the ILP Model

Variable	Description
W_i	CPU capacity of physical node $i \in N_p$
$C_{(i,j)}$	Routing capacity of physical link $(i,j) \in E_p$
w_s	CPU capacity requirement of virtual node $s \in N_l$
$c_{(s,t)}$	Routing link capacity of virtual link $(s,t) \in E_l$
H_k	Amount of reparation resources available for substrate nodes at stage $k \in K$
h_i	Amount of reparation resources required for physical node $i \in N_p$ to be repaired
R_k	Amount of reparation resources available for substrate links at stage $k \in K$
$r_(i,j)$	Amount of reparation resources required for physical link $(i,j) \in E_p$ to be repaired
g_d	Amount of reparation resources required for DC $d \in D_p$ to be repaired
G_k	Amount of reparation resources available for DCs at stage $k \in K$
$\alpha_{1,v}^k$	Weight for slice v at stage $k \in K$
$\alpha_{2,(s,t)}^k$	Weight of virtual link $(s,t) \in \overline{E}_l$ at stage $k \in K$

Objective: Maximizing the *accumulative weighted number of recovered slices and virtual links* (AWRSL).

$$\max \quad \sum_{k=0}^{K}\sum_{v\in V}\alpha_{1,v}^{k}S_{v}^{k}+\sum_{k=0}^{K}\sum_{(s,t)\in\overline{E}_l}\alpha_{2,(s,t)}^{k}\overline{\beta}_{(s,t)} \tag{1}$$

B. Constraints

We first describe the constraints for the *Any Re-embedding* strategy, being it the most generic case where all reconfiguration actions are allowed.

1) Node embedding constraints: Eqn. (2) ensures that no more than one virtual node of slice v is embedded to the same physical node i at stage k. Eqns (3) and (4) ensure that a virtual node s is embedded to exactly one physical node while satisfying the location requirement (i.e., to a physical node among the set G_s). Eqn. (5) ensures that the computing capacity of the physical nodes is not exceeded.

$$\sum\nolimits_{s\in N_l^v} m_{i,s}^k \le 1 \quad \forall v\in V, i\in N_p \tag{2}$$

$$\sum\nolimits_{i\in G_s} m_{i,s}^k = 1 \quad \forall s\in N_l \tag{3}$$

$$\sum\nolimits_{i\in (N_p-G_s)} m_{i,s}^k = 0 \quad \forall s\in N_l \tag{4}$$

$$\sum\nolimits_{s\in N_l} w_s * m_{i,s}^k \le W_i \quad \forall i\in N_p \tag{5}$$

2) Link embedding with flow constraints: Eqn. (6) is the flow conservation constraint. Eqns. (7) and (8) ensure loops are avoided while embedding a virtual link (s,t). Eqn. (9) ensures that the virtual link (s,t) and (t,s) are embedded to same physical links (in opposite directions). Eqn. (10) ensures that a virtual link (s,t) cannot be embedded to both directions of a physical link and Eqn. (11) is the link capacity constraint.

$$\sum\nolimits_{(i,j)\in E_p} q_{(i,j)}^{(s,t),k} - \sum\nolimits_{(j,i)\in E_p} q_{(j,i)}^{(s,t),k} = m_{i,s}^k - m_{i,t}^k \quad \forall i\in N_p, (s,t)\in E_l \tag{6}$$

$$\sum\nolimits_{(i,j)\in E_p} q_{(i,j)}^{(s,t),k} \le 1 \quad \forall i\in N_p, (s,t)\in E_l \tag{7}$$

$$\sum\nolimits_{(i,j)\in E_p} q_{(i,j)}^{(s,t),k} \le 1 \quad \forall j\in N_p, (s,t)\in E_l \tag{8}$$

$$q_{(i,j)}^{(s,t),k} - q_{(j,i)}^{(t,s),k} = 0 \quad \forall (i,j)\in E_p, (s,t)\in E_l \tag{9}$$

$$q_{(i,j)}^{(s,t),k} + q_{(j,i)}^{(s,t),k} \le 1 \quad \forall (i,j)\in E_p, (s,t)\in E_l \tag{10}$$

$$\sum\nolimits_{(s,t)\in E_l} c_{(s,t)} * q_{(i,j)}^{(s,t),k} \le C_{(i,j)} \quad \forall (i,j)\in E_p \tag{11}$$

3) Network state constraints: Eqn. (12) ensures that the physical link (i,j) is working (not failing) only when node i, node j, and the link (i,j) are working. Eqn. (13) ensures a virtual link *(s,t)* is recovered only when all the physical links it is embedded on are working. Eqn. (14) and (15) ensure that a DC provides content to slices only if the DC is working.

Eqn. (12) contains "AND" operation, which can be linearized as in [21].

$$\beta_{(i,j)}^k = \sum\nolimits_{l=0}^{k} p_{(i,j)}^k \wedge \sum\nolimits_{l=0}^{k} z_i^k \wedge \sum\nolimits_{l=0}^{k} z_j^k \quad \forall (i,j)\in E_p \tag{12}$$

$$\overline{\beta}_{(s,t)}^k * q_{(i,j)}^{(s,t),k} \le \beta_{(i,j)}^k \quad \forall (s,t)\in E_l, (i,j)\in E_p \tag{13}$$

$$A_d^k \le \sum\nolimits_{i\in D_p} (m_{i,d}^k * \sum\nolimits_{l=0}^{k} b_d^l) \quad \forall d\in D_l \tag{14}$$

$$Q_{d,n}^k \le A_d^k \quad \forall v\in V, n\in N_l^v, d\in D_l^v \tag{15}$$

4) Connectivity constraints under substrate failures: The constraints for NC are Eqns (16), (17), (18), and (21). The constraints for CC are Eqns (16), (17), (19), (20), and (22). Eqn. (16) is the flow constraint to check whether a DC can reach other virtual nodes. Eqn. (17) ensures that a DC can not provide content to virtual node n through a failed virtual link. Eqns (19) and (20) check whether a virtual node n can reach at least one DC. Eqns (18) and (21) ensure that a slice guarantees NC only if the slice is connected and at least one DC d can provide content to all nodes. Eqn. (22) ensures a slice guarantees CC only if all its nodes can reach content from at least one DC.

$$\sum_{(s,t)\in E_l^v} f_{(s,t)}^{(d,n),k} - \sum_{(t,s)\in E_l^v} f_{(t,s)}^{(d,n),k} = \begin{cases} Q_{(d,n)}^k & s=d \\ -Q_{(d,n)}^k & s=n \\ 0 & otherwise \end{cases} \quad \forall n\in N_l^v, d\in D_l^v, s\in N_l^v : d\neq n \tag{16}$$

$$f_{(s,t)}^{(d,n),k} \le \overline{\beta}_{(s,t)}^k \quad \forall n\in N_l^v, d\in D_l^v, (s,t)\in E_l^v : d\neq n \tag{17}$$

$$\overline{Q}_d^k \le Q_{d,n}^k \quad \forall v\in V, n\in N_l^v, d\in D_l^v \tag{18}$$

$$\overline{Q}_n^k \ge Q_{d,n}^k \quad \forall v\in V, n\in N_l^v, d\in D_l^v \tag{19}$$

$$\overline{Q}_n^k \le \sum\nolimits_{d\in D_l^v} Q_{d,n}^k \quad \forall v\in V, n\in N_l^v \tag{20}$$

$$S_v^k \le \sum\nolimits_{d\in D_l^v} \overline{Q}_d^k \quad \forall v\in V, d\in D_l^v \tag{21}$$

$$S_v^k \le \overline{Q}_n^k \quad \forall v\in V, n\in N_l^v \tag{22}$$

5) Recovery and resource constraint: Eqns (23)–(25) ensure that at each stage k, the resources used to recover links, nodes, and DCs cannot exceed the corresponding total resources up to that stage. The variables related to links, nodes, and DCs by fixing $p^0_{(i,j)}$, z^0_i, and b^0_d are initialized to 0. Eqns (26)–(28) ensures that the physical links, nodes, and DCs are only repaired once.

$$\sum_{l=0}^{k} \sum_{(i,j)\in E_f} p^l_{(i,j)} r_{(i,j)} \leq \sum_{l=1}^{k} R_l \tag{23}$$

$$\sum_{l=0}^{k} \sum_{i\in N_f} z^l_i * h_i \leq \sum_{l=1}^{k} H_l \tag{24}$$

$$\sum_{l=0}^{k} \sum_{d\in D_f} b^l_d * g_d \leq \sum_{l=1}^{k} G_l \tag{25}$$

$$\sum_{k\in K} p^k_{(i,j)} \leq 1 \quad \forall (i,j) \in E_p \tag{26}$$

$$\sum_{k\in K} z^k_i \leq 1 \quad \forall i \in N_f \tag{27}$$

$$\sum_{k\in K} b^k_d \leq 1 \quad \forall d \in D_p \tag{28}$$

C. Extension of ILP Model to Different Recovery Strategies

We now extend the ILP formulation of the *Any Re-embedding* strategy of Sec. IV-B to the other three recovery strategies (Fixed Initial Embedding, Link Re-embedding and Node Re-embedding) by enforcing node and/or link embedding constraints for all stages $k \in K$.

1) Fixed Initial Embedding Strategy: We denote the initial embedding as $\overline{m}_{i,s}$ and $\overline{q}^{(s,t)}_{(i,j)}$. To keep initial node and link embedding, Eqns (29) and (30) are added, which ensure that the embedding is fixed at all stages.

$$m^k_{i,s} = \overline{m}_{i,s} \quad \forall i \in N_p, s \in N_l \tag{29}$$

$$q^{(s,t),k}_{(i,j)} = \overline{q}^{(s,t)}_{(i,j)} \quad \forall (i,j) \in E_p, (s,t) \in E_l \tag{30}$$

2) Link Re-embedding Strategy: This strategy allows link re-embedding but constrains the initial node embedding, which is obtained by Eqn. (29), but not Eqn. (30).

3) Any Re-embedding(γ_1, γ_2) Strategy: This strategy allows node and link re-embedding and link reconfiguration actions but limits node and DC reconfiguration to γ_1 and γ_2), respectively. So, we impose Eqns 31 and 32 as follows:

$$\sum_{i\in(N_p-N_d)} \sum_{s\in N_l} (m^k_{i,s} - m^k_{i,s} * m^{k-1}_{i,s}) \leq \gamma_1 \tag{31}$$

$$\sum_{i\in N_d} \sum_{s\in N_l} (m^k_{i,s} - m^k_{i,s} * m^{k-1}_{i,s}) \leq \gamma_2 \tag{32}$$

where $m^k_{i,s} - m^k_{i,s} * m^{k-1}_{i,s}$ equals 1 if the embedding of node s in stage k is different from the stage $k-1$.

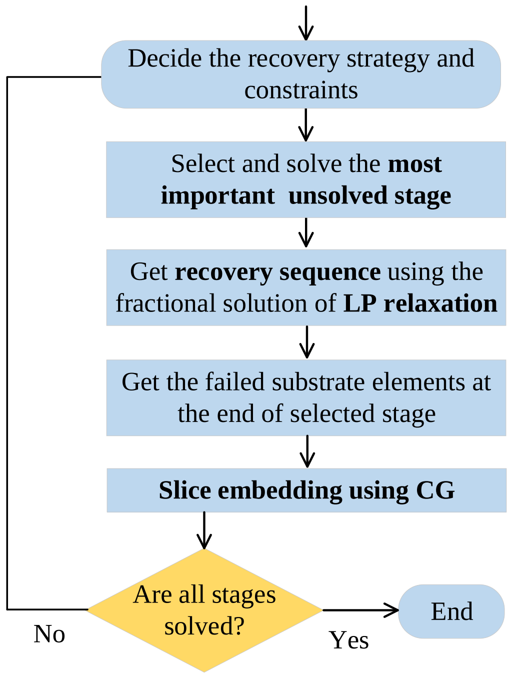

Decide the recovery strategy and constraints
Select and solve the **most important unsolved stage**
Get **recovery sequence** using the fractional solution of **LP relaxation**
Get the failed substrate elements at the end of selected stage
Slice embedding using CG
Are all stages solved?
No
Yes
End

Fig. 2: Sequence diagram of the two-phases progressive slice recovery algorithm.

V. Two-phases Progressive Slice Recovery Algorithm

Given the computational limitation of the previous formulation, in this section, we introduce a two-phases progressive slice recovery (2-phase PSR) algorithm to solve the PSR problem. First, we overview the overall proposed 2-phase PSR algorithm. Then, we propose a deterministic rounding procedure for the first phase, i.e., the recovery-sequence phase, and we discuss how to use the column generation algorithm (CG) to speed up the second phase, i.e., the slice embedding phase. Finally, we summarize the resulting 2-phase PSR algorithm using a detailed pseudocode.

A. Generalized Procedure for Recovery

Given the hardness of the PSR problem (see Section III-C), we divide the problem into two parts, 1) recovery sequence and 2) slice embedding. Fig. 2 shows the sequence diagram of the two-phases progressive slice recovery algorithm. The weight of a stage at stage $k \in K$ is defined as the sum of weights of all the slices at stage k. The algorithm iteratively solves the most important unsolved stage, which has the biggest weight among unsolved stages.

Recovery sequence. First, we solve the LP relaxation of the ILP model in Section IV separately for each stage and obtain the fractional solution of the relaxed LP model in polynomial time. Second, we propose a deterministic rounding based procedure to approximate the recovery sequence from the solution of the LP relaxation. We solve the LP relaxation, not in a single round, but stage by stage, because a gradual stage by stage approach provides a more close approximation of the recovery sequence at initial stages. More specifically,

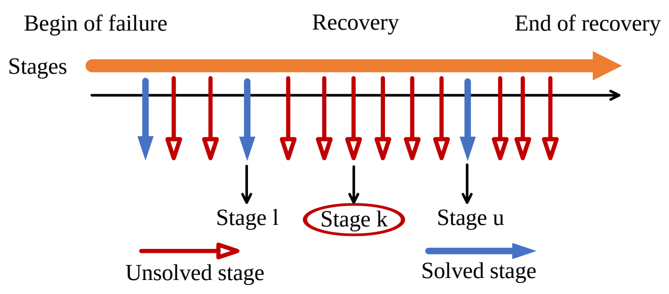


Fig. 3: An example of selecting stage to solve.

note that, if we perform LP relaxation, the obtained solution will return, during initial few stages, that all virtual links are repaired, even though not all the physical elements over which these virtual links are mapped are repaired; hence, in the following stages, since LP relaxation wrongly considers virtual links are already repaired, the model will not repair other physical components. Consequently, a single-round LP relaxation does not provide useful information for recovery sequence at following stages.

Slice embedding. We use CG[1] to solve the slice embedding in the damaged physical network. The order for solving different stages is important since it restricts the recovery and embedding process of other stages. Each slice may have different weights in different stages, and the weight measures the importance of the slice at that stage. We start solving stages with greater weights, which configures priority to more important slices. When solving the embedding in a specific stage, the node embedding needs to be consistent with the (re)-embedding constraints across each stage. For *Fixed Initial Embedding*, we add Eqns (29) and (30) from Section IV-C. For *Link Re-embedding*, we add Eqn. (29). No additional constraints are needed for *Any Re-embedding*. For *Any Re-embedding(γ_1, γ_2)*, if we have already solved the embedding in stage k_1 and we want to solve the stage $k_2 > k_1$, the node embedding must satisfy the Eqns (33) and (34). Set $M_n^{k_1}$ and $M_d^{k_1}$contain embedding of normal nodes and nodes with DC at stage k_1. $\sum_{(i,s)\in M_n^{k_1}}(1-m_{i,s}^{k_2})$ is the number of embedding of normal nodes that are disrupted due to reconfiguration. $\sum_{(i,s)\in M_d^{k_1}}(1-m_{i,s}^{k_2})$ is the number of embedding of DC nodes that are disrupted due to reconfiguration.

$$\sum\nolimits_{(i,s)\in M_n^{k_1}}(1-m_{i,s}^{k_2}) \leq (k_2-k_1)*\gamma_1 \qquad (33)$$
$$\sum\nolimits_{(i,s)\in M_d^{k_1}}(1-m_{i,s}^{k_2}) \leq (k_2-k_1)*\gamma_2 \qquad (34)$$

B. Deterministic Rounding based Recovery Sequence Determination

The deterministic rounding-based procedure consists of two steps. First, based on chosen version of ILP (i.e., based on the chosen recovery strategy), we solve the corresponding LP relaxation. Second, we propose Algorithm 1, a deterministic rounding based procedure, to obtain the recovery sequence for substrate nodes, links, and DCs. The variable $p_{(i,j)}^k, z_i^k, b_d^k$ can be seen as the likelihood to repair the corresponding links, nodes, and DCs at stage k. At each stage k^*, the substrate components that are allowed to be repaired are the ones that are recovered between k_l and k_u (line 1, 7, 13), which are the nearest solved stage before and after stage k^*, respectively.

As shown in Fig. 3, we select stage k to solve, for which the failed links are denoted as E_f. Since we already solved stage u since it is more important, the recovered links between stage k and stage u can not be the failed links at stage u since it is not used until stage u. We denote the failed links, nodes, DCs at stage l as E_f, N_f, and D_f, which is the failed components in the current phase. Besides, the failed links, nodes, DCs at stage u are denoted as E_u, N_u, and D_u. Since the links are not recovered until stage u, the links that are allowed to be repaired at current phase are $E_f - E_u$ (line 7). The algorithm recovers the links according to their values of $\overline{p}_{(i,j)}^1$ until the remained resource cannot recover any links (line 3, 4, 5). The nodes and DCs are recovered using similar approach as for links (line 7-18).

Algorithm 1 Deterministic rounding procedure for recovery sequence

```
Input: $E_f, N_f, D_f, E_u, N_u, D_u, \overline{R}_1, \overline{H}_1, \overline{G}_1, \overline{p}^k_{(i,j)}, \overline{z}^k_i, \overline{b}^k_i$
Output: $E_f, N_f, D_f, p^k_{i,j}, z^k_i, b^k_i$
 1: $E_r \leftarrow \{(i,j)|r_{(i,j)} \leq \overline{R}_1, (i,j) \in E_f - E_u\}$
 2: while $E_r \neq \emptyset$ do
 3:     Let $(i^*,j^*) \leftarrow argmax_{(i,j)\in E_r}\{\overline{p}^1_{(i,j)}|r_{(i,j)} \leq \overline{R}_1\}$
 4:     $p^1_{(i^*,j^*)} \leftarrow 1, \overline{R}_1 \leftarrow \overline{R}_1 - r_{(i^*,j^*)}, E_f = E_f - \{(i^*,j^*)\}$

 5:     $E_r \leftarrow \{(i,j)|r_{(i,j)} \leq \overline{R}_1, (i,j) \in E_f - E_u\}$
 6: end while
 7: $N_r \leftarrow \{i|h_i \leq \overline{H}_1, i \in N_f - N_u\}$
 8: while $N_r \neq \emptyset$ do
 9:     Let $i^* \leftarrow argmax_{i\in N_r}\{\overline{z}^1_i|h_i \leq \overline{H}_1\}$
10:     $z^1_{i^*} \leftarrow 1, \overline{H}_1 \leftarrow \overline{H}_1 - h_{i^*}, N_f \leftarrow N_f - \{i^*\}$
11:     $N_r \leftarrow \{i|h_i \leq \overline{H}_1, i \in N_f - N_u\}$
12: end while
13: $D_r \leftarrow \{d|g_d \leq \overline{G}_1, d \in D_f - D_u\}$
14: while $D_r \neq \emptyset$ do
15:     Let $d^* \leftarrow argmax_{d\in D_r}\{\overline{b}^1_d|g_d \leq \overline{G}_1\}$
16:     $b^1_{(i^*,j^*)} \leftarrow 1, \overline{G}_1 \leftarrow \overline{G}_1 - g_{d^*}, D_f = D_f - \{d^*\}$
17:     $D_r \leftarrow \{d|g_d \leq \overline{G}_1, d \in D_f - D_u\}$
18: end while
```

C. CG based Slice Embedding in Damaged Network

After having determined which failed substrate components have been recovered at the end of each stage in the previous phase, CG performs slice embedding using an auxiliary graph in [29]. An example of the auxiliary graph is reported in Fig. 4, where every virtual node s is connected to the substrate nodes in $G[s]$ with dotted lines. The set of auxiliary links is denoted as A_e. Finding an embedding for virtual link (A,B) is equivalent to finding a path from A to B in the auxiliary graph. CG requires to identify a reduced master problem (RMP) and a pricing problem, which are solved iteratively. We keep solving

[1]CG is a widely used methodology to solve large scale ILP problems. It has been shown to be scalable for large topologies on the VNE problem [28].

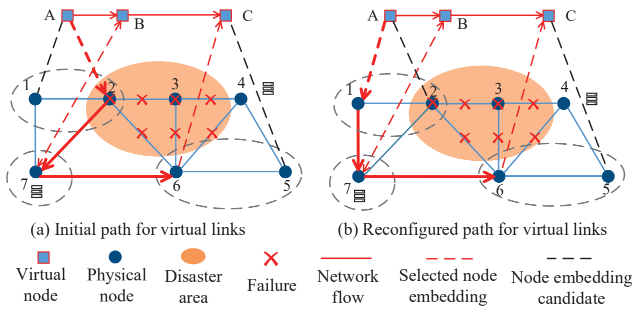


Fig. 4: Auxiliary graph in CG

the LP relaxation of RMP and search for a path related to an improving non-basic variable for RMP, then add the variable to RMP until no path can improve the objective value of the RMP. At last, we solve the RMP with integer constraints to get the embedding of slices.

1) Process for slice embedding decision: The RMP generates embedding for only one stage. We denote the set of paths for virtual link (s,t) as $P_{(s,t)}$. Variable $\overline{q}^{p,k}_{(s,t)}$ equals to 1 if the virtual link (s,t) uses the path p at stage k. The parameter $\delta^{p,(i,j)}_{(s,t)}$ equals to 1 if the path p for the virtual link (s,t) contains link $(i,j) \in E_p \cup A_e$, denoted as $\overline{E}_p$. The RMP problem aims to maximize the weighted working slices and virtual links, according to the following objective function:

$$\max \sum_{v \in V} \alpha^k_{1,v} * S^k_v + \sum_{(s,t)\in \overline{E}_l} \sum_{p \in P_{(s,t)}} \alpha^k_{2,(s,t)} * \overline{q}^{p,k}_{(s,t)} \tag{35}$$

Moreover, the RMP contains some constraints from Sec. IV, namely *Node embedding constraints*, *Network state constraints* and *Connectivity constraints under substrate failures* except Eqn. (13). The following constraints are then added:

$$A^k_d \le \sum\nolimits_{i \in D_p} (m^k_{i,d} * b^k_d) \quad \forall d \in D_l \tag{36}$$

$$\sum\nolimits_{(s,t)\in \overline{E}_l} \sum\nolimits_{p \in P_{(s,t)}} c_{s,t} * \delta^{p,(i,j)}_{(s,t)} * \overline{q}^{p,k}_{(s,t)} \le C_{(i,j)} * \beta^k_{(i,j)} \quad \forall (i,j) \in E_p \tag{37}$$

$$\sum\nolimits_{p \in P_{(s,t)}} \overline{q}^{p,k}_{(s,t)} \le 1 \quad \forall (s,t) \in \overline{E}_l \tag{38}$$

$$\sum\nolimits_{p \in P_{(s,t)}} \delta^{p,(i,s1)}_{(s,t)} * \overline{q}^{p,k}_{(s,t)} \le m^k_{i,s1} \quad \forall (i,s1) \in A_e \tag{39}$$

$$\sum\nolimits_{p \in P_{(s,t)}} \overline{q}^{p,k}_{(s,t)} \ge \overline{\beta}^k_{(s,t)} \quad \forall (s,t) \in \overline{E}_l \tag{40}$$

Eqn. (36) ensures that a DC is available only if the physical node that it is embedded to is working. Eqn. (37) ensures that if the physical link (i,j) is working, the capacity of the logical links embedded over it should not exceed its capacity. $P_{s,t}$ is the set of possible paths for the embedding of virtual link (s,t). Eqn. (38) ensures there is one path selected for a virtual link. Eqn. (39) ensures that virtual link (s,t) is not embedded to the path that contains node embedding $(i,s1)$ if virtual node $s1$ is not embedded to physical node i. Variable b^k_d is fixed to 0 if a DC d fails. Eqn. (40) ensures that a virtual link *(s,t)* works only when one working path is selected for it.

2) Embedding path selection: The pricing problem aims to find paths that improve the objective value of the master problem and adds the corresponding path to the set $P_{s,t}$. The variable $\phi_{1,i,j}$, $\phi_{2,s,t}$, $\phi^{s1,i}_{3,s,t}$, $\phi_{4,s,t}$ are the dual variables of the Eqns (37), (38), (39), and (40). The variable $U_{(s,t)}$ equals to 1 if the pricing problem generates a path for virtual link (s,t) while the $\overline{f}^{(s,t)}_{(i,j)}$ equals to 1 if the generated path for virtual link (s,t) includes the physical link (i,j). Set $\overline{E}^v_l$ denotes the unidirectional link in slice v. A path can improve the objective value if the reduced cost (RC) of the corresponding variable $\overline{q}^{p,k}_{(s,t)}$ is negative. The objective function of the pricing problem is to minimize the RC of $\overline{q}^{p,k}_{(s,t)}$, which is listed as follows:

$$\begin{aligned} \min & - \sum_{(i,j)\in E_p} \sum_{(s,t)\in \overline{E}_l} \phi_{1,i,j} * c_{(s,t)} * \overline{f}^{(s,t)}_{(i,j)} \\ & - \sum_{(s,t)\in \overline{E}_l} \phi_{2,s,t} * U_{(s,t)} - \sum_{(s,t)\in \overline{E}_l} \sum_{(s1,i)\in A_u} \phi^{s1,i}_{3,s,t} * \overline{f}^{(s,t)}_{(i,j)} \\ & - \sum_{(s,t)\in \overline{E}_l} \phi_{4,s,t} * U_{(s,t)} + \sum_{(s,t)\in E_l} \alpha^k_{2,(s,t)} * U_{(s,t)} \end{aligned} \tag{41}$$

The constraints of the pricing problem ensure that the path found for virtual link $(s,t) \in \overline{E}_l$ is feasible and does not go through failed substrate components. Eqn. (42) ensures that the pricing problem only generates a path for one virtual link. Eqn. (43) ensures that the generated path does not contain failed nodes or links. Eqn. (44) is the flow conservation constraint for each virtual link. Eqn. (45) ensures that at most one path is selected for each virtual link. Eqn. (46) ensures that a path does not contain both directions of a physical link. Eqn. (47) ensures that for each virtual link $(s,t) \in \overline{E}_l$, the two virtual node s and t are embedded to different physical nodes.

$$\sum\nolimits_{(s,t)\in E_{l,u}} U_{(s,t)} = 1 \quad \forall (s,t) \in \overline{E}_l \tag{42}$$

$$\overline{f}^{(s,t)}_{(i,j)} = 0 \quad \forall (s,t) \in \overline{E}_l, \forall (i,j) \in E_f \parallel i \in N_f \parallel j \in N_f \tag{43}$$

$$\sum\nolimits_{(i,j)\in \overline{E}_p} \overline{f}^{(s,t)}_{(i,j)} - \sum\nolimits_{(j,i)\in \overline{E}_p} \overline{f}^{(s,t)}_{(i,j)} = \begin{cases} U_{(s,t)} & i = s \\ -U_{(s,t)} & i = t \\ 0 & otherwise \end{cases} \quad \forall (s,t) \in \overline{E}_l \tag{44}$$

$$\sum\nolimits_{(s,t)\in \overline{E}_l} \sum\nolimits_{(s,i)\in A_e} \overline{f}^{(s,t)}_{(s,i)} \le 1 \tag{45}$$

$$\overline{f}^{(s,t)}_{(i,j)} + \overline{f}^{(s,t)}_{(j,i)} \le 1 \quad \forall (s,t) \in \overline{E}_l, (i,j) \in E_p \tag{46}$$

$$\sum\nolimits_{(i,j)\in E_p} \overline{f}^{(s,t)}_{(i,j)} \ge 1 \quad \forall (s,t) \in \overline{E}_l \tag{47}$$

D. *Scalable Two-phases Progressive Slice Recovery Algorithm*

We now illustrate the details of the proposed scalable progressive slice recovery algorithm through Algorithm 2, which integrates the two phases, namely *recovery sequence* (line 5-13) and *slice embedding* (line 14-16), and solves all stages iteratively.

All the strategies for PSR problem take same inputs except that *Any Re-embedding(γ_1, γ_2)* takes two more inputs γ_1 and γ_2 to define the reconfiguration action. The variable μ_k equals to 0 if the recovery sequence and slice embedding are not determined at stage k. Stage 0 is the stage that no resources are used and stage k_{max} is the last stage for recovery. The failed links, nodes, and DCs at stage k are represented as $N_{f,k}$, $E_{f,k}$, and $D_{f,k}$, respectively.

Algorithm 2 Two-phases progressive slice recovery algorithm

```
Input: $N_f, D_f, E_f, \alpha^k_{1,v}, \alpha^k_{2,(s,t)}(, \gamma_1, \gamma_2)$
Output: $p^k_{i,j}, z^k_i, b^k_i, m^k_{i,s}, q^{(s,t),k}_{(i,j)}$
    Initialisation : $\mu_k = 0, iter = 1$
 1: $\mu_{-1} = 1, \mu_{k_{max}+1} = 1, \overline{K} \leftarrow 0, 1$
 2: $E_{f,-1} \leftarrow E_f, N_{f,-1} \leftarrow N_f, D_{f,-1} \leftarrow D_f$
 3: $E_{f,k_{max}+1} \leftarrow \emptyset, N_{f,k_{max}+1} \leftarrow \emptyset, D_{f,k_{max}+1} \leftarrow \emptyset$
 4: while $iter < k_{max} + 1$ do
 5:     Find $k^*$ such that $\sum_{v \in V} \alpha^{k^*}_v = max\{\sum_{v \in V} \alpha^k_v | 0 < k < k_{max}, \gamma_k \neq 1\}$
 6:     $\mu_{k^*} \leftarrow 1, iter \leftarrow iter + 1$
 7:     Find $k_l$ such that $k_l < k^*$, $\mu_{k_l} = 1$ and $\mu_k = 0, \forall k_l < k < k^*$
 8:     Find $k_u$ such that $k_u > k^*$, $\mu_{k_u} = 1$ and $\mu_k = 0, \forall k^* < k < k_{max}$
 9:     $E_f \leftarrow E_{f,k_l}, N_f \leftarrow N_{f,k_l}, D_f \leftarrow D_{f,k_l}$
10:     $E_u \leftarrow E_{f,k_u}, N_u \leftarrow N_{f,k_u}, D_u \leftarrow D_{f,k_u}$
11:     $\overline{H}_1 \leftarrow \sum_{k=k_l}^{k^*} H_k, \overline{R}_1 \leftarrow \sum_{k=k_l}^{k^*} R_k, \overline{G}_1 \leftarrow \sum_{k=k_l}^{k^*} G_k$
12:     Solve the relaxed LP model with constraints (1)-(28) and constraints in Section V-A, get $\overline{p}^k_{i,j}, \overline{z}^k_i, \overline{b}^k_i$
13:     Call Algorithm 1 to get recovery sequence
14:     $R_{k^*} \leftarrow \overline{R}_{k^*}, H_{k*} \leftarrow \overline{H}_{k^*}, G_{k^*} \leftarrow \overline{G}_{k^*}$
15:     $E_{f,k^*} \leftarrow E_f, N_{f,k^*} \leftarrow N_f, D_{f,k^*} \leftarrow D_f$
16:     Solve slice embedding at stage $k^*$ using CG in Section V-C.
17: end while
```

The Algorithm 2 initializes the failed component at stage -1 and k_{max+1}, which serves as the bound for failed components (line 1-3). In each iteration, after determining the unsolved stage k^* with the biggest weight (line 5), the algorithm finds the stages k_l and k_u to serve as the bounds (line 7, 8). Then, the failed components at stage k_l are used as the failed component at stage k^*, which is represented as E_f, N_f, and D_f as the one in the ILP model (line 9). The failed components at stage k_u must be a subset of the failed component at stage k^*, which are E_u, N_u, and D_u (line 10). The stages for the relaxed LP contains two stages (stage 0 and 1), which is denoted as $\overline{K}$. $\overline{H}_1, \overline{R}_1$, and $\overline{G}_1$ are the resources for nodes, links, and DCs at stage k^*, which is used in the relaxed LP as the resources in stage 1 (line 11, 12). After calling the deterministic rounding procedure in Algorithm 1 to get the recovery sequence (line 13), we save the remained resources and the failed component at stage k^* (line 14, 15) and solve the slice embedding at stage k^* (line 16).

VI. Numerical Results

In this section, we first validate the performance of the proposed 2-phase PSR algorithm, comparing its performance to that of ILP in terms of optimality gap and execution time. Then, we compare different recovery strategies in large network scenarios.

A. Simulation Setup

We implemented the ILP formulations using AMPL (A Mathematical Programming Language) and CPLEX MIP solver. The simulations are performed on a workstation with Intel(R) Core(TM) i5-8400 CPU (6 cores @ 2.80GHz) processor and 32768 MB of memory. In our evaluations, we consider two substrate network topologies, a small network topology (*small*) (7 nodes and 11 bidirectional links) and a large network topology (*large*) (the USnet topology in [21], 24 nodes and 43 bidirectional links). Similar to [17], capacities of substrate links and nodes for the small and large network topologies are set to generic values following a uniform distribution in [20, 80] and [80, 100], respectively, and capacity requirements of virtual nodes and links are also uniformly distributed in [1, 5]. To compare ILP and the proposed 2-phase PSR algorithm, we consider the small substrate network topology (shown in Fig. 1) with 10 slices embedded to it, each consisting of 3 virtual nodes in which two are DCs (DC nodes), and 3 virtual links. We consider that every link, node, and DC in the 7-node substrate network fail with a probability of 30%. To extend our evaluation, we compare the different recovery strategies considering the 2-phase PSR algorithm in the large substrate topology with 20 embedded slices. The number of virtual nodes in each slice is an integer value uniformly distributed in [5, 10] in which two are DCs (DC nodes). We consider three different cases of failure rate where every link, node, and DC fail with a probability of 30%, 40%, and 50%, respectively.

B. Comparison of ILP and the 2-phase PSR Algorithm

Now we compare the performance of ILP and 2-phase PSR algorithm. Specifically, we compare four different scenarios, namely, ILP with guaranteed NC (NC-ILP), 2-phase PSR algorithm with guaranteed NC (NC-P), ILP with guaranteed CC (CC-ILP), and 2-phase PSR algorithm with guaranteed CC (CC-P) for all the four recovery strategies reported in Table I.

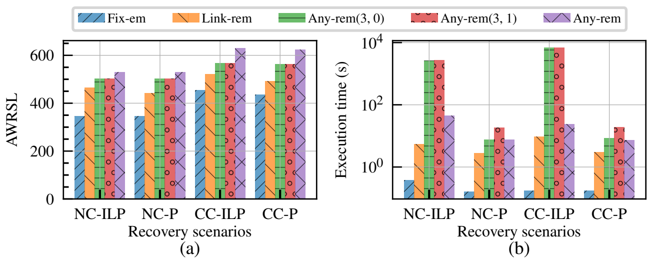


Fig. 5: Comparison of ILP and the proposed 2-phase PSR algorithm.

Fig. 5(a) shows the objective function, i.e., the *accumulative weighted number of recovered slices and virtual links*

(AWRSL of the four scenarios for the four recovery strategies). Recapping, in terms of AWRSL, *Fixed Initial Embedding* reaches the lowest value while *Any Re-embedding* has the highest value. *Any Re-embedding*(γ_1, γ_2) can significantly increase AWRSL compared with *Link Re-embedding* by just reconfiguring few nodes. For example, AWRSL of *Link Re-embedding* in NC-ILP is 465, which can be increased to 502 by allowing reconfiguration action of 2 normal nodes and 1 DC node in *Any Re-embedding*(γ_1, γ_2). The objective of *Any Re-embedding* in NC-ILP is 530, which is 5.6% larger than *Any Re-embedding*(γ_1, γ_2). Moreover, results show that CC has around 10% higher objective value than NC in all strategies for both the ILP and the 2-phase PSR algorithm. Regarding the execution time (Fig. 5(b)), results show that the proposed 2-phase PSR algorithm has a significantly lower execution time than ILP, especially for *Any Re-embedding*(γ_1, γ_2) and *Any Re-embedding*. For instance, CC-ILP takes 6798.10 seconds to get the solution of *Any Re-embedding*$(3, 1)$ while CC-P requires only 19.13 seconds. This savings in execution time are achieved for very small optimality gaps between ILP and the proposed 2-phase PSR algorithm. The 2-phase PSR algorithm reaches optimum in all cases except that of *Link Re-embedding*, which is around 5%. The optimality gap for CC is less than 6% for both of *Fixed Initial Embedding* and *Link Re-embedding* while for both of *Any Re-embedding* and *Any Re-embedding*(γ_1, γ_2) the optimality gaps are lower than 2%.

In conclusion, the proposed 2-phase PSR algorithm outperforms ILP in terms of execution time, paying off only a small optimality gap compared with ILP. Moreover, results show how AWRSL can be significantly increased by reconfiguring only few nodes or by guaranteeing CC in place of NC.

C. *Evaluation on a Large Topology with Network Connectivity (NC)*

To extend our comparison of the different recovery strategies, we contrast their AWRSL (obtained with our proposed 2-phase PSR algorithm) on the *large* substrate network topology. Fig. 6(a) and (b) plot the AWRSL and the number of recovered slices at stage 0 (i.e., the "AWRSL at stage 0") with guaranteed NC for the different recovery strategies with respect to different percentages of substrate failures ranging between 20% and 40%, respectively. Results show that, in all cases, the re-embedding constraints and the allowed reconfiguration actions play a decisive role in the recovery. Specifically, the AWRSL of *Link Re-embedding* makes up only 36.3%, 73.6%, and 89.1% of those of *Any Re-embedding* at substrate failure rates of 20%, 30%, and 40%, respectively (note that, for a higher number of failed network elements, the gain of embedding reconfigurability decreases as the number of working substrate elements that can be re-embedded over are smaller). Moreover, results also show that as the percentage of substrate failure becomes larger, *Any Re-embedding*(γ_1, γ_2) strategy becomes increasingly competitive compared to *Fixed Initial Embedding* and *Link Re-embedding* as, even with a small number of reconfiguration action of normal nodes and DC nodes allowed, the objective values of *Any Re-embedding(5,2)* strategy are

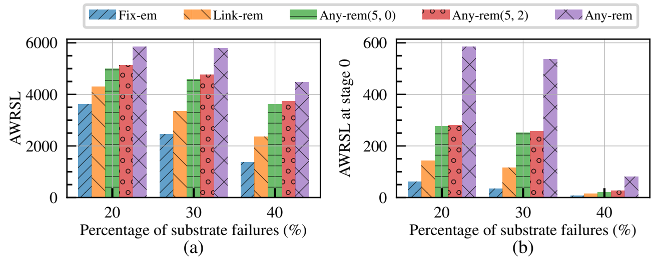


Fig. 6: Recovery outcome at different substrate network failure rate of the different recovery strategies with NC.

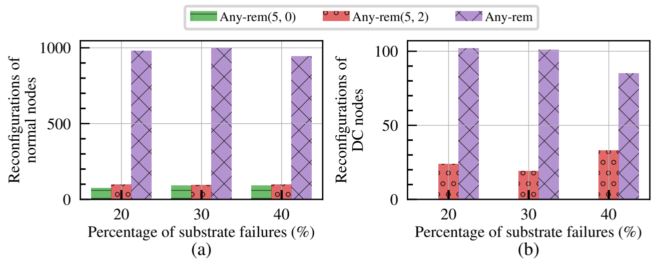


Fig. 7: Number of reconfigurations vs. substrate network failure rate for the different recovery strategies with NC.

19.5%, 42.7%, and 58.5% larger than *Link Re-embedding*. Fig. 6(b) shows the objective value at stage 0, before any substrate component is repaired[2]. Results show the advantage *Any remapping* recovery strategy provides compared to different recovery strategies. This implies that disaster relief can happen much faster right after the failure by allowing different reconfiguration actions.

Fig. 7 shows the total number of reconfigurations for normal nodes and DC nodes at all stages. *Fixed Initial Embedding* and *Link Re-embedding* are not shown since they do not have any reconfigurations of nodes. When the substrate failure rate is 20%, the number of reconfiguration actions of normal nodes and DC nodes in *Any Re-embedding* is more than 10 times and 4 times larger than *Any Re-embedding*$(5, 2)$, however, the AWRSL of *Any Re-embedding* is only 1.14 times that of *Any Re-embedding*$(5, 2)$, demonstrating the capability of *Any Re-embedding*(γ_1, γ_2) to reach promising trade-off among numbers of node re-embedding and recovery performance. In conclusion, by selecting the number of allowed reconfigurations of nodes at each stage carefully, the network operator can achieve satisfying recovery performance with acceptable number of reconfigurations.

Fig. 8 shows the recovered capacity of nodes and links in all slices per stage under different substrate failure rate while enforcing NC. Recovering the capacity as early as possible means recovering the slices as early as possible. In all cases, *Any Re-embedding* reaches the maximum capacity earliest while *Fixed Initial Embedding* is the latest to recovering all the capacities. This implies that the network operator can recover the virtual node and link capacity earlier if the number of allowed reconfiguration of nodes is larger. Besides, as

[2]Note that different strategies shows different values of the objective since they allow different reconfiguration actions

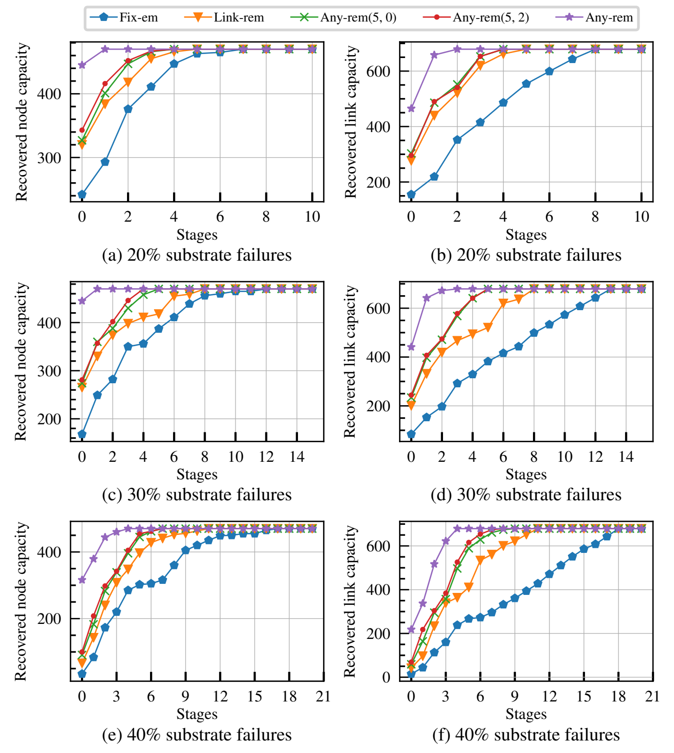


Fig. 8: Substrate node and link utilization per stage under different substrate network failure rate for the different recovery strategies with NC.

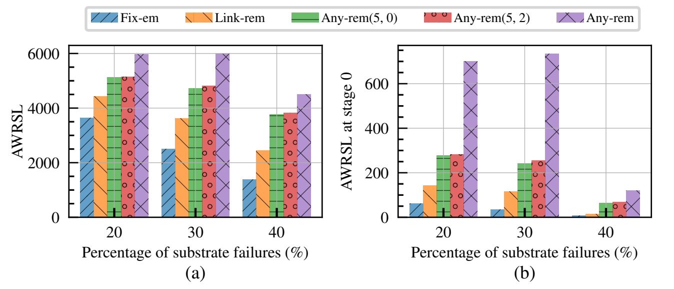


Fig. 9: Recovery outcome at different substrate network failure rate of the different recovery strategies with CC.

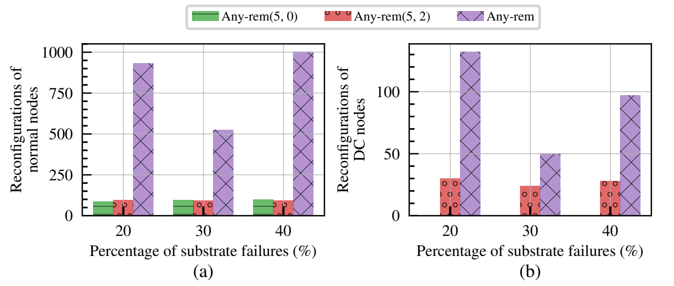


Fig. 10: Number of reconfigurations vs. substrate network failure rate for the different recovery strategies with CC.

the percentage of substrate failures increases, the number of allowed reconfiguration also becomes increasingly important.

D. Evaluation on a Large Topology with Content Connectivity (CC)

We now compare the performance of the different recovery strategies considering CC. Fig. 9(a) and (b) plot the AWRSL and the number of recovered slices at stage 0 (i.e., the "AWRSL at stage 0") with guaranteed CC for the different recovery strategies with respect to different percentages of substrate failures ranging between 20% and 40%, respectively. Even with a small number of reconfiguration action of normal nodes and DC nodes allowed, the objective values of *Any Re-embedding(5,2)* strategy enforcing CC are 16.1%, 32.8%, and 56.3% larger than Link Re-embedding.Moreover, Fig. 10 shows the recovered capacity of nodes and links in all slices per stage under different substrate failure rate while enforcing CC. Results show that the recovery outcome and the number of reconfigurations when enforcing CC are consistent with those when enforcing NC. Specifically, when enforcing CC, AWRSL can be improved by a value ranging from 1% to 8% with respect to NC.

VII. Conclusion

With the increased adoption of 5G services (slices) that are characterized by high-reliability requirements (e.g., service for remote diagnosis or smart factory), a fast slice recovery is crucial in case of failures affecting the network infrastructure. In this paper, we proposed, for the first time to the best of our knowledge, the Progressive Slice Recovery (PSR) problem. We systematically classified different recovery strategies for rapid slice recovery when allowing re-embedding and reconfiguration of slices and when not, while ensuring Network Connectivity (NC) or Content Connectivity (CC), and proved its NP-hardness. We first formulated the PSR problem mathematically through an ILP model, which maximizes the *accumulative weighted number of recovered slices and virtual links* (AWRSL). Then, to obtain an approximation of the best recovery sequence, we relaxed the integer constraints and used the deterministic rounding technique to obtain the recovery sequence of substrate components. After that, we performed the embedding of network slice through Column Generation (CG). After bench-marking the proposed CG-based approach to the ILP, we performed exhaustive simulations considering different failure rates of the substrate network. Our illustrative numerical results show the relative performance of eight different slice recovery strategies in terms of the accumulative weighted number of recovered slices. Results show that recovery strategies that allow any re-embedding achieve an AWRSL 300% higher than recovery strategies with limited or no reconfiguration. Nevertheless, our numerical results show that even with a limited number of re-embeddings allowed, the recovery outcome can be significantly improved. Moreover, results show that by guaranteeing CC instead of NC, recovery outcome can be improved by up to 56.3%.

References

[1] S. Zhang, "An overview of network slicing for 5G," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 111–117, Jun. 2019.
[2] P. Caballero, A. Banchs, G. d. Veciana, and X. Costa-Pérez, "Multi-Tenant Radio Access Network Slicing: Statistical Multiplexing of Spatial Loads," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 3044–3058, Oct. 2017.

[3] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *IEEE International Conference on Computer Communications (INFOCOM)*, Apr. 2009, pp. 783–791.

[4] N. Shahriar, S. Taeb, S. R. Chowdhury, M. Tornatore, R. Boutaba, J. Mitra, and M. Hemmati, "Achieving a fully-flexible virtual network embedding in elastic optical networks," in *IEEE International Conference on Computer Communications (INFOCOM)*, Apr. 2019, pp. 1756–1764.

[5] Q.-T. Luu, S. Kerboeuf, A. Mouradian, and M. Kieffer, "A Coverage-Aware Resource Provisioning Method for Network Slicing," *IEEE/ACM Transactions on Networking*, vol. 28, no. 6, pp. 2393–2406, Dec. 2020.

[6] S. Liao, J. Wu, S. Mumtaz, J. Li, R. Morello, and M. Guizani, "Cognitive Balance for Fog Computing Resource in Internet of Things: An Edge Learning Approach," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.

[7] F. Fossati, S. Moretti, P. Perny, and S. Secci, "Multi-Resource Allocation for Network Slicing," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1311–1324, Jun. 2020.

[8] C. Sexton, N. Marchetti, and L. A. DaSilva, "On Provisioning Slices and Overbooking Resources in Service Tailored Networks of the Future," *IEEE/ACM Transactions on Networking*, vol. 28, no. 5, pp. 2106–2119, Oct. 2020.

[9] N. Shahriar, R. Ahmed, S. R. Chowdhury, M. M. A. Khan, R. Boutaba, J. Mitra, and F. Zeng, "Virtual network embedding with guaranteed connectivity under multiple substrate link failures," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 1025–1043, Feb. 2020.

[10] G. Le, S. Ferdousi, A. Marotta, S. Xu, Y. Hirota, Y. Awaji, M. Tornatore, and B. Mukherjee, "Survivable virtual network mapping with content connectivity against multiple link failures in optical metro networks," *Journal of Optical Communications and Networking*, Jul. 2020.

[11] X. Li, S. Huang, S. Yin, Y. Zhou, M. Zhang, Y. Zhao, J. Zhang, and W. Gu, "Design of k-node (edge) content connected optical data center networks," *IEEE Communications Letters*, vol. 20, no. 3, pp. 466–469, Mar. 2016.

[12] J.-P. Vasseur, M. Pickavet, and P. Demeester, *Network recovery: protection and restoration of optical, SONET-SDH, IP, and MPLS*. Elsevier, 2004.

[13] J. Rak and D. Hutchison, *Guide to disaster-resilient communication networks*. Springer, 2020.

[14] M. M. A. Khan, N. Shahriar, R. Ahmed, and R. Boutaba, "Multi-path link embedding for survivability in virtual networks," *IEEE transactions on network and service management*, vol. 13, no. 2, pp. 253–266, 2016.

[15] B. Mukherjee, M. F. Habib, and F. Dikbiyik, "Network adaptability from disaster disruptions and cascading failures," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 230–238, May 2014.

[16] S. R. Chowdhury, R. Ahmed, M. M. Alam Khan, N. Shahriar, R. Boutaba, J. Mitra, and F. Zeng, "Dedicated protection for survivable virtual network embedding," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 913–926, Dec. 2016.

[17] L. Gao and G. N. Rouskas, "Virtual network reconfiguration with load balancing and migration cost considerations," in *IEEE International Conference on Computer Communications (INFOCOM)*, Apr. 2018.

[18] Z. Zhou, T. Lin, and K. Thulasiraman, "Survivable cloud network design against multiple failures through protecting spanning trees," *Journal of Lightwave Technology*, vol. 35, no. 2, pp. 288–298, 2017.

[19] J. Wang, C. Qiao, and H. Yu, "On progressive network recovery after a major disruption," in *IEEE International Conference on Computer Communications (INFOCOM)*, Apr. 2011.

[20] S. Ciavarella, N. Bartolini, H. Khamfroush, and T. La Porta, "Progressive damage assessment and network recovery after massive failures," in *IEEE International Conference on Computer Communications (INFOCOM)*, May 2017.

[21] S. Ferdousi, M. Tornatore, F. Dikbiyik, C. U. Martel, S. Xu, Y. Hirota, Y. Awaji, and B. Mukherjee, "Joint progressive network and datacenter recovery after large-scale disasters," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2020.

[22] C. Ma, J. Zhang, Y. Zhao, M. F. Habib, S. S. Savas, and B. Mukherjee, "Traveling repairman problem for optical network recovery to restore virtual networks after a disaster [invited]," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 7, no. 11, pp. B81–B92, Nov. 2015.

[23] N. Shahriar, R. Ahmed, S. R. Chowdhury, A. Khan, R. Boutaba, and J. Mitra, "Generalized recovery from node failure in virtual network embedding," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 261–274, Jun. 2017.

[24] S. Ferdousi, M. Tornatore, S. Xu, Y. Awaji, and B. Mukherjee, "Slice-aware service restoration with recovery trucks for optical metro-access networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2019.

[25] M. Pourvali, K. Liang, F. Gu, H. Bai, K. Shaban, S. Khan, and N. Ghani, "Progressive recovery for network virtualization after large-scale disasters," in *2016 International Conference on Computing, Networking and Communications (ICNC)*, Feb. 2016, pp. 1–5.

[26] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *IEEE International Conference on Computer Communications (INFOCOM)*, Apr. 2006.

[27] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: a survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[28] R. Mijumbi, J. Serrat, J.-L. Gorricho, and R. Boutaba, "A path generation approach to embedding of virtual networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 3, pp. 334–348, Sep. 2015.

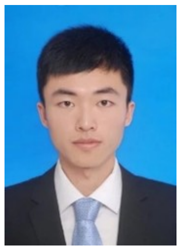
[29] Q. Hu, Y. Wang, and X. Cao, "Resolve the virtual network embedding problem: A column generation approach," in *IEEE International Conference on Computer Communications (INFOCOM)*, Apr. 2013.

Qiaolun Zhang is pursuing the master's degree in School of Electronic Information and Electrical Engineering in Shanghai Jiao Tong University, Shanghai, China. He is selected to a double master degree program between Shanghai Jiao Tong University and Politecnico di Milano. He received his M.S. degree in computer science and engineering from the Politecnico di Milano in 2020. He has participated in many projects in lab of next generation networking. His research interests include Machine Learning, Internet of things(IoT), Optimization, etc.

Omran Ayoub is a post-doctoral research fellow at the Department of Electronics, Information and Bioengineering (DEIB) at Politecnico di Milano, Milan, Italy, where he obtained his M. Sc. degree in 2015 and his Ph.D. degree in 2019. His research interests are in the field of optical networking, cloud computing and optimization of communications networks. Dr. Ayoub is author of more than 30 papers in the area of communication networks, published in international journals and conference proceedings.

Jun Wu (S'08-M'12) received the Ph.D. degree in information and telecommunication studies from Waseda University, Japan, in 2011. He was a Post-Doctoral Researcher with the Research Institute for Secure Systems, National Institute of Advanced Industrial Science and Technology (AIST), Japan, from 2011 to 2012. He was a Researcher with the Global Information and Telecommunication Institute, Waseda University, Japan, from 2011 to 2013. He is currently a professor of School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China. He is also the vice dean of Institute of Cyber Science and Technology and vice director of National Engineering Laboratory for Information Content Analysis Technology, Shanghai Jiao Tong University, China. He is the chair of IEEE P21451-1-5 Standard Working Group. He has hosted and participated in a lot of research projects including National Natural Science Foundation of China (NFSC), National 863 Plan and 973 Plan of China, Japan Society of the Promotion of Science Projects (JSPS), etc. His research interests include the advanced computing, communications and security techniques of software-defined networks (SDN), information-centric networks (ICN) smart grids, Internet of Things (IoT), 5G/6G, etc., where he has published more than 140 refereed papers. He has been the Track Chair of VTC 2019, VTC 2020 and the TPC Member of more than ten international conferences including ICC, GLOBECOM, etc. He has been a Guest Editor of the IEEE Sensors Journal, Sensors, ICT Express. He is an Associate Editor of the IEEE Access, IEEE Networking Letters.

Francesco Musumeci (S'11-M'12) received the Ph.D. degree in Information Engineering from Politecnico di Milano, Italy, in 2013, where he is currently an Assistant Professor with the Department of Electronics, Information and Bioengineering since 2016. His current research interests include Machine-Learning-assisted networking, design, optimization and performance analysis of optical networks, 5G, NFV/SDN and network disaster resilience. Dr. Musumeci is author of more than 90 papers in the area of communication networks, published in international journals and conference proceedings, and is co-winner of three best paper awards from IEEE sponsored conferences. He used to serve as a TPC member and/or reviewer for several IEEE/OSA conferences as well as IEEE/OSA and Elsevier journals since 2010.



Gaolei Li (S'16-M'21) received the B.S. degree in electronic information engineering from Sichuan University, Chengdu, China, and PhD degree in Cyber Security from Shanghai Jiao Tong University, Shanghai, China. From Oct. 2018 to Sep. 2019, he visited the Muroran Institution of Technology, Muroran, Japan, supported by the China Scholarship Council Program. Now, he is an Assistant Professor in School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include 5G/6G security, machine learning security and blockchain. He has received best paper awards from the IEEE ComSoc CSIM Committee and Chinese Association for Cryptologic Research, and a student travel grant award for IEEE Globecom. He is the reviewer of IEEE TDSC, TII, ACM TOIT, and TPC member for IEEE iThings 2016-2019, IEEE ICC 2018-2021, IEEE ISPA 2020, ScalCom 2020&2021.

Massimo Tornatore (S'03–M'06–SM'13) is currently an Associate Professor with the Department of Electronics, Information, and Bioengineering, Politecnico di Milano, where he received a Ph.D. degree in 2006. He also holds an appointment as Adjunct Professor at University of California, Davis, USA and as visiting professor at University of Waterloo, Canada. His research interests include performance evaluation, optimization and design of communication networks (with an emphasis on the application of optical networking technologies), cloud computing, and machine learning application for network management. In these areas, he co-authored more than 400 peer-reviewed conference and journal papers (with 19 best paper awards), 2 books and 1 patent. He is a member of the Editorial Board of IEEE Communication Surveys and Tutorials, IEEE Communication Letters, Springer Photonic Network Communications, and Elsevier Optical Switching and Networking. He is active member of the technical program committee of various networking conferences such as INFOCOM, OFC, ICC, and GLOBECOM.