deleted

Shuopeng Li

Survivable Services Oriented Protection level-aware Virtual Network Embedding

Shuopeng LI[1,2], Mohand Yazid SAIDI[1] and Ken CHEN[1]

Abstract—Network virtualization permits the creation of several logical networks (virtual networks) on one shared physical network referred as the substrate network. To protect a network against single substrate link failures, fast local reroute is preferred. With the reservation of backup resources, the flows are switched quickly from primary to backup paths upon substrate link failure to ensure service continuity.
Due to the difficulty of primary and backup mappings, most of works in the literature separates the mapping of primary virtual network from the setting of backup paths. Although this approach optimizes primary resources, it can lead to inefficient protection since the existence of backup paths depends on the selected primary paths. In this paper, we propose a framework for protection-level-aware virtual network embedding which minimizes the risks of unrecoverable failures. With our propositions, the primary paths are selected among those which can be fully protected, if there is no such path, then we take the least vulnerable links in order to minimize the failure probability. For primary mapping, we propose a flexible on-line backup verification-based heuristic and a fast backup pre-verification-based heuristic. With the first heuristic, the backup path feasibility is verified on-line for each potential primary link, whereas we pre-compute for each substrate link the optimized set of backup tunnels in advance with the second heuristic. Simulations show that our propositions significantly reduce the substrate link failure impact on virtual networks, at the price of a slight decrease of the primary acceptance ratio.

Index Terms—Network virtualization, reliability, protection, survivability, routing, backup paths, virtual network embedding

1 INTRODUCTION

Network virtualization [1] facilitates the creation of logical networks (referred as virtual network, VN) on a shared physical network (referred as substrate network, SN). Service providers build and manage their VN-based services in an on-demand way without having to touch the underlying infrastructure that provides the physical resources. This process is called *Virtual Network Embedding* (*VNE*). With VNE, Infrastructure Providers (InPs) can react to demands of users in an agile manner by dynamically creating virtual networks, with an efficient use of their physical resources.

With the fifth generation cellular network technology (5G), various network use cases are defined and should be satisfied to meet the multiple needs and constraints of future services such as broadband access, massive internet of things, high mobility, extreme real-time communications and ultra-reliable communications. In order to provide an effective platform for supporting these services, 5G is adopting network slicing which enables the creation of tailored virtual networks. In this way, to provide a reliable service to users, it is sufficient and often better to create only one fault-tolerant VN rather than protecting the entire physical infrastructure.

In parallel with the development and deployment of 5G, we expect widespread use and emergence of new real-time services such as VoIP, video conferencing, telemedicine, autonomous driving, and more. These services are sensitive to communication disruptions and therefore need to operate on reliable networks capable of quickly repairing failures to ensure service continuity.

We recall that networks fail for various reasons, such as device failure, software attacks, natural disasters, etc. If the survivability against failures is not guaranteed, a single failure may affect several VNs and results in severe penalties for service providers. Recall that failure survivability or reliability aims to ensure service continuity even upon failures. It involves a range of issues concerning the design of networks [2]. Two main techniques are often used to cope with failures [3]: restoration and protection. Restoration is a re-active approach, where no computation is performed before the failure occurs. No extra resources are used. On the contrary, protection is a pro-active approach, which pre-computes backup paths before any failure. Generally, protection reserves resources for the backup paths in order to guarantee enough resources and ensure service continuity upon failure.

For virtual networks, the survivability is provided in 2 stages by protection or restoration [4]. When a VN request arrives, service provider first determines a virtual network embedding solution that often optimizes the link and node resources (generally, bandwidth and delay for links, and CPU and memory for nodes) [5]; after this, some restoration or protection method [3] is applied. Although this approach does not optimize the overall resources (primary and backup resources), it has the merit of optimizing the flows that actually use resources. In other words, flows that follow the primary paths are optimized outside the rare and short periods of failures. However, such approach can lead to the difficulty to find backup paths since protection capabilities

• [1]*L2TI, Institut Galilé, Université Paris 13, Villetaneuse, France*
[2]*BeiJing University of Technology, BeiJing, China*
E-mail: lishuopeng1025@qq.com, {saidi,ken.chen}@univ-paris13.fr

are not taken into account at the primary embedding step.

Other survivable VNE methods [6], [7] compute jointly the primary and backup mappings to optimize the overall allocated resources. Although these VNEs increase the number of protected VNs, they lead to non optimal routing most of the time, since the flows follow the primary paths outside the rare periods of failures. These methods are more suitable for offline computations (they consume a lot of time for computations) and rather preferable for re-optimizing resource allocations.

To address the precedent issues, we propose here a novel survivable VNE that significantly improves the reliability by combining the failure avoidance-oriented approach and protection capability-aware technique. In this paper, we use and favor protection over failure avoidance-oriented approach which is the only technique explored in [8]. In this way, recovery from failures is guaranteed as long as the protection resources are sufficient. In our proposition, we separated for each new request the substrate links into two subsets: (i) *protectable* links which can be protected by backup paths and (ii) *unprotectable* links which cannot be protected due to insufficient resources on backup paths. Accordingly, the primary flows are computed such that they follow the optimal substrate paths which privilege the use of protectable links, then the least *vulnerable* links among unprotectable ones. Note that only unprotectable links are considered as *vulnerable*: higher is the failure probability of a given link, more vulnerable is that link. With our proposition, the backup path selection is taken into consideration at the step of primary mapping to ensure the best flow protection. Instead of focusing only on resource optimization at the step of primary mapping, we propose here to (1) maximize the reliability (i.e. reduce VN failures) by selecting the protectable and least vulnerable links which (2) improve the resource utilization.

Under the assumption of single failures, we formulate the problem of survivable virtual link mapping (we assumed that the node mapping is already done) as an Integer Linear Programming (ILP) problem aiming to minimize the VN failure probability and enhance the resource utilization. To solve the precedent problem, we first defined for each VN request a substrate link cost that depends on both backup path existence and link failure probability. We then proposed two heuristics, both based on approached Steiner trees minimizing the precedent cost. With our first heuristic, the existence of backup paths is verified on-line for each substrate and virtual links whereas we pre-compute in advance the optimal backup tunnels for each substrate link, according to the max-flow solution, with our second heuristic.

The rest of this paper is organized as follows. Section 2 provides an overview of the related work. Section 3 describes our assumptions and network model, and gives an example illustrating the basic idea of our proposal. Sections 4 and 5 present respectively our ILP formulation and heuristics for the problem resolution. The evaluation results are shown in Section 6. Section 7 concludes this paper.

2 Related Work

Due to difficulty of joint primary and backup mappings, most of works in literature adopted the two stages-based survivable VNE which separates the primary VNE from the backup mapping. Despite this separation, the two resulting sub-problems often remain NP-hard. Indeed, VNE problem without protection (with its numerous variants) is proved to be NP-hard [5] [9] [10] and protection of one path (and thus one virtual link) is also NP-hard. Similarly, joint optimization of primary and backup paths is NP-hard [11].

In this section, we first describe works dealing with VNE without protection and then focus on survivable VNE.

2.1 Virtual network embedding

As said previously, VNE problem (with its various variants) is NP-hard. For instance, the problem of mapping unconnected virtual nodes can be reduced to the Bin packing problem that is NP-hard. Similarly, when the nodes are already embedded, the mapping of links verifying the bandwidth constraints can be reduced to the multicommodity problem which is NP-hard for unsplittable traffics[1]. Finally, joint embedding of links and nodes is also NP-hard and can be reduced to the multiway separator problem [10].

In a multi-operator network, Houidi et al. [13] proved that assigning the virtual nodes to only 2 InPs can be reduced to the well known MAX-2-SAT NP-hard problem.

Heuristics have been proposed in literature to solve various variants of VNE problem. These heuristics usually reduce the solution space by eliminating some dimensions or by exploring only the promising areas. For instance, the constraints related respectively to the topology and to the bandwidth are eliminated in [13] and [14] to solve VNE. In [15], an approach based on subgraph isomorphism detection is presented. Another model in [16] applies the Markov Random Walk to rank nodes and then embeds links and nodes by using back-tracking strategy based on breadth-first search.

In [17], the node and link mappings are performed in two separate stages. In the first stage, the virtual nodes are embedded on substrate nodes which are more likely to reduce the substrate path lengths. In the second stage, the virtual links are mapped on substrate paths verifying the constraints and minimizing the costs. Unlike the first stage which depends on both the link and node properties, the second one only explores the link properties.

2.2 Survivable virtual network embedding

Network reliability can be achieved with the combination of failure avoidance oriented approach and protection methods. The failure avoidance approach aims to determine solely a primary embedding which minimizes the risk of failures by selecting substrate components among the most reliable. Such approach does not thus use backup paths. The protection methods consist in pre-computing and often preconfiguring backup routes which are to be activated in case of failures.

With regards to the failure avoidance oriented approach, Li and al. [8] proved that VNE minimizing the failure probability without admission control can be reduced to a Steiner Tree problem (known as NP-hard). When the bandwidth is taken into account, various heuristics combining approximated Steiner trees are proposed.

1. The problem is tractable for fractional flows [12].

Concerning the protection, various approaches are proposed in the literature for virtual network survivability. Two main approaches can be distinguished: protection against virtual node failures and protection against virtual link failures.

2.2.1 Protection against virtual node failures

To protect against the failures of virtual nodes which correspond to the virtual machines, the dedicated protection (1+1) is generally preferred to the node migration. With 1+1 protection, the virtual nodes are replicated and deployed on both primary and backup substrate nodes. For each protected node, the traffic is sent both on primary and backup routes to reach respectively the primary and backup nodes.

In [18], the problem of 1+1 node protection minimizing the overall resources (bandwidth, CPU, etc.) is formulated with mixed integer linear programming. In this formulation, each primary virtual node is interconnected to all its adjacent primary virtual nodes and their backup ones, in addition of its corresponding backup node. In this way, any number of failures affecting the primary nodes could be tackled without any service disruption. Due to the high complexity of the precedent mixed integer linear programming, the authors proposed a polynomial time heuristic that reduces the solution space by (i) mapping the virtual nodes and links in separate stages, (ii) adding some new practical constraints related to the geographic location and to the limitation of the number of site candidates, and finally (iii) deleting the connections between each primary node and its backup one. In [19], the authors proposed another similar approach in which the virtual nodes and links are embedded separately. For backup virtual node embedding, the substrate nodes which can support a maximum number of virtual nodes are privileged. Once the virtual nodes are mapped, the primary and backup nodes are connected with the use of trees to reduce the bandwidth consumption.

Although, the 1+1 protection minimizes the recovery time, it present a high cost limiting its deployment and resulting in the resource wastage. Moreover, to reduce the bandwidth consumption, the precedent approaches share the bandwidth between the primary and backup paths which is hard to achieve in practice. Indeed, primary and backup traffics should traverse the common links at the same time to permit the bandwidth sharing. This results in the use of unidirectional trees for primary and backup routes.

2.2.2 Protection against virtual link failures

Like the dedicated protection (1+1) which can deal with both the link and node failures, the shared protection (1:N) is available for any type of failures. However, for acceptable recovery time, the shared protection is often restricted to the protection of virtual links since the shared protection of nodes requires virtual node migration which increases substantially the recovery time.

Two substrate paths are selected for the embedding of one protected virtual link: primary path and backup path. With the single failure assumption, all backup paths that protect against different risks (substrate links and nodes) can *share* their resource allocations. In this way, a large amount of resources can be saved at the cost of increased recovery time which now includes detection and re-routing times.

Protection can be provided on virtual layer or substrate layer. In virtual layer, the original VN is often reinforced with protection capacity. In [20] and [21], backup virtual nodes and links are added to VN topology in order to provide protection against node failures. For link protection, [7] proposes to add virtual backup links to the VN before embedding it by solving the multicommodity flow problem. In this way, each virtual link is mapped to two disjoint pre-computed paths (primary and backup) so that the bandwidth and penalty for bandwidth violation are optimized. This method applies the dedicated protection which increases and wastes the bandwidth in a failure-free case. In [22], the additional backup resources are optimized by searching for disjoint end-to-end backup shortest paths.

In substrate layer, fast reroute allows each substrate link to deal with the failure locally. In [7], the authors propose a heuristic that provides survivability for VNs in three steps: (i) node embedding, (ii) link embedding and (iii) selection of backup detours. With this restoration approach, a set of backup detours protecting each substrate link are pre-computed and activated at the failure of that link so that the penalty incurred for violating the bandwidth constraint is minimized. In [23], a splittable share protection problem is formulated with pre-selected backup candidates. Backup topology on substrate network is simplified in [24]. The p-cycle technique combined with a column generation optimization model is adopted in [25] to provide protection against node and link failures.

A probabilistic model is proposed in [26] to deal with Shared Risk Link Group (SRLG) failures. We recall that an SRLG corresponds to a group of links sharing a same physical component. When the latter fails, all the links of SRLG also fail simultaneously. Authors in [26] formulated the problem as a Non-Linear Programming problem and propose several approximation algorithms to solve it. In [27], relative disjoint paths are generated to maximize the VN reliability while enhancing the bandwidth allocation. After associating to each link a weight based on bandwidth and risk scores, the authors propose, for VN embedding, to first determine the primary routing reducing the weight and then, add redundancies as and when necessary to satisfy the requirements of users. Instead of optimizing the resource allocation by selecting the optimal set of substrate links which maximizes the reliability, this approach merely adds redundant paths until the desired level of reliability is achieved.

Similarly, in elastic optical networks, the authors in [28] first define a novel metric to measure the VN failure probability before proposing an embedding algorithm that decreases the defined metric. With this algorithm, each virtual link is mapped to the most reliable couple of paths (primary and backup paths). As all the possible mapping orders are computed, this approach is not scalable and unpractical for large networks.

Although most of the link protection methods presented in this paper provide survivability with acceptable running time, they do not guarantee optimal protection. Indeed, the methods separating primary embedding of VNs from the backup embedding do not take into account the protection

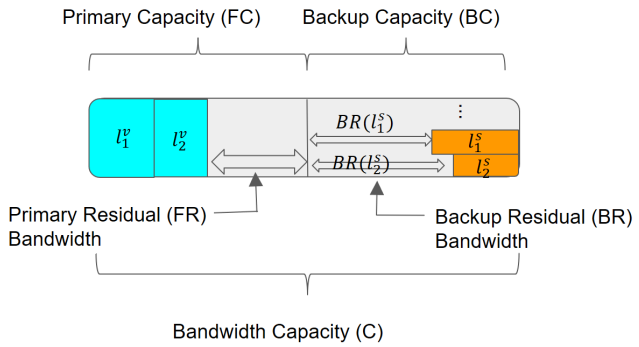
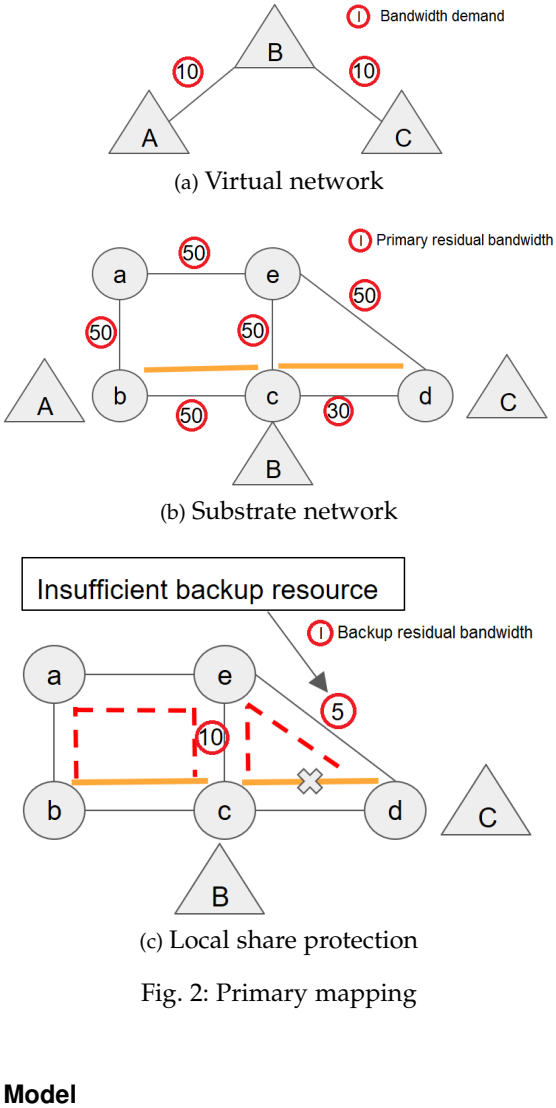


Fig. 1: Bandwidth separation



(a) Virtual network

(b) Substrate network

(c) Local share protection

Fig. 2: Primary mapping

capabilities at the step of primary mapping. Only the time consuming methods that jointly map the primary and backup VNs generally optimize the protection. To cope with the precedent issues, we propose in the next sections, a novel survivable VNE that take into account the protection capabilities at the step of primary mapping to maximize the reliability.

3 Assumptions, Model and Problem Illustration

In this paper, we treat the problem of virtual link mapping that maximizes the reliability by combining the protection capability-aware technique with the failure avoidance-oriented approach. Before formulating the problem in the next section, we describe below the assumptions and network model we adopted in this paper and then, we illustrate the problem through an example.

3.1 Assumptions

For our study, we adopted the following assumptions.

3.1.1 Pre-established node mapping step

As we are only focusing on virtual link mapping maximizing the reliability, we assumed that the step of node mapping is already completed. This step can be accomplished by applying various algorithms like [12] [14] [17] [19].

We denote by $M()$ the node mapping function, so $M(A)$ implies the substrate node on which the virtual node A is embedded.

3.1.2 link-related assumptions

Each link is associated with a failure probability that can be fixed according to various parameters such as the physical characteristics of the link (coaxial cable, optical fiber, etc.), its geographic location (land, sea, hot or cold region, etc.), its use frequency and load, etc. The failure probabilities of links could also be determined statistically by monitoring the network infrastructure.

For our study, we adopted the commonly practice assumption of independent link failure probabilities. In addition, to enable the bandwidth sharing and save the resources, we assumed single link failures. This last assumption means that only one failure can occur at a given time.

3.2 Model

3.2.1 Substrate network

The substrate network is modeled as a connected undirected graph $G^S(N^S, L^S)$, where N^S is the set of substrate nodes and L^S is the set of substrate links. Each substrate node $n^s \in N^S$ is associated with various metrics (CPU, memory, etc.) and constraints (geographic location, etc.). Similarly, each substrate link l^s is associated with a bandwidth capacity $C(l^s)$ and a failure probability P_{l^s}. Note that we do not consider other attributes, like delay or financial cost.

3.2.2 Virtual network

The virtual network is also modeled as a connected undirected graph $G^V(N^V, L^V)$, where N^V is the set of virtual nodes and L^V is the set of virtual links. Each virtual node $n^v \in N^V$ is associated with a demand related to various metrics (CPU, memory, etc.). Each $l^v \in L^V$ is associated with a bandwidth demand $bw(l^v)$. In addition, each virtual network G^V has a lifetime $t(G^V)$.

3.2.3 Primary and backup resource separation

We separated the bandwidth capacity C_{l^s} on each link l^s into two pools: primary bandwidth capacity denoted by FC_{l^s} and backup bandwidth capacity denoted by BC_{l^s}. This bandwidth separation into two pools allows operators

to decide, macroscopically, the amount of bandwidth dedicated to primary flows and protection.

As shown in Figure 1, primary capacity is occupied by primary links ($l_1^v, l_2^v, etc.$) whereas backup capacity is dedicated for backup paths protecting against failures of substrate links ($l_1^s, l_2^s, etc.$). The primary residual bandwidth of substrate link l^s is denoted by FR_{l^s}. Since the backup resources are shared, a substrate backup link bl^s could protect against the failure of several substrate links l^s. For each l^s, the backup residual resource is defined as $BR_{bl^s}(l^s)$.

The ratio between the primary bandwidth and the total bandwidth capacity on a link is denoted by τ ($\tau = \frac{FC_{l^s}}{C_{l^s}}$). This parameter τ can be dynamically adjusted per link, by monitoring the networking status for instance.

3.3 Primary embedding and protection example

A primary mapping example is shown in Figure 2. The virtual network consists of 3 nodes $\{A, B, C\}$ and 2 virtual links $\{A-B, B-C\}$ (see Figure 2a). Both virtual links demand 10 units of bandwidth. The substrate network (see Figure 2b) is composed of 5 nodes and 6 links. Each substrate link l^s is labeled with the corresponding primary residual bandwidth FR_{l^s}. We assume that the node mapping is: $M(A) = b$, $M(B) = c$ and $M(C) = d$. If the optimization criterion is primary bandwidth minimization without reliability (protection) consideration, a possible mapping solution is given by solid lines in Figure 2b (i.e., virtual link $A-B$ and $B-C$ are respectively mapped to the substrate paths $b-c$ and $c-d$).

To ensure service continuity and reduce disruption time upon failure, we adopted in this paper the local share protection. For each substrate link l^s, a backup path is computed to deal with its possible failure. We precise that the backup paths are bidirectional. Hence, the extremity nodes of the backup paths should correspond to the extremity nodes of the protected link in order to permit local and rapid switching between the primary and backup paths upon failure.

With the assumption of single failures, the resources of backup paths which protect against different failures are shared on substrate links to optimize the resource utilization.

For our example in Figure 2, we assumed that the backup residual bandwidth to protect against the failure of substrate link $c-d$ is 10 units for all the substrate links except for substrate link $e-d$ where it corresponds to 5 units (see Figure 2c). Without admission control (i.e., without bandwidth constraint verification), link $b-c$ can be protected by the backup path $b-a-e-c$ whereas link $c-d$ can be protected by the backup path $c-e-d$. In this way, the two determined backup paths $b-a-e-c$ and $c-e-d$ can share their resource allocations on link $e-c$ since these paths protect against different failure risks (link $b-c$ and link $c-d$ respectively).

Despite the bandwidth sharing, we see that with the bandwidth constraint verification, neither the backup path $c-e-d$ nor any other backup path can be used to protect against the failure of link $c-d$ since such path should traverse link $e-d$ whose available backup resources (5 units) are not sufficient to satisfy the protection request

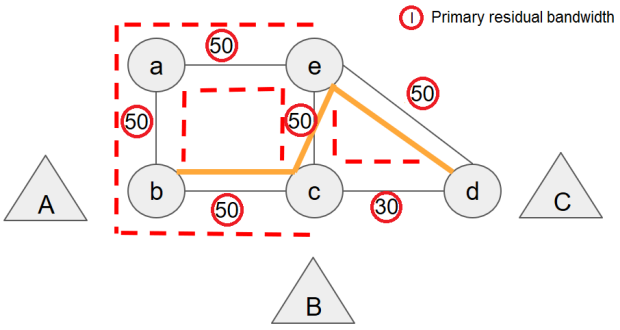


Fig. 3: Primary mapping with backup consideration and local share protection

(10 units). Thus, to ensure complete link protection, the selected primary mapping should be modified to bypass the link $c-d$ and privilege the use of protectable links. In our example, the virtual link $B-C$ should be mapped on the substrate path $c-e-d$ as shown in Figure 3. In this manner, all the primary substrate links can be protected. Indeed, link $b-c$ which embeds the virtual link $A-B$ is protected by the backup path $b-a-e-c$ while links $c-e$ and $e-d$ which embed the virtual link $B-C$ are protected with the use of the backup paths $c-b-a-e$ and $e-c-d$ respectively. Note that the primary bandwidth constraints are also verified as shown in Figures 2b and 3 where the labels correspond to the amounts of primary residual bandwidth on the corresponding links.

From the previous example, we deduce that the protection capabilities depend on the selected primary paths. In the following sections, we study the problem of reliability maximization. We first formulate the problem with integer linear programming (Section 4) and give two effective heuristics to solve it (Section 5).

4 Our Proposition

We recall that we are solely dealing with link mapping, by assuming that the node mapping has been completed.

Our proposition consists of a combination of link protection and failure avoidance, where the link protection is preferred to failure avoidance. We formulate it as an Integer Linear Problem (ILP) in which the VNE solution minimizing the primary bandwidth resource is preferred among all the solutions maximizing the reliability.

In order to protect as much as possible the primary links, backup path constraints should be added. A link $bl^s \in L^S$ can be used to protect a virtual link l^v against the failure risk l^s if it verifies the backup bandwidth constraints, i.e. if $BR_{bl^s}(l^s) - bw(l^v) \geq 0$.

4.1 Formulation

We formulate the survivable VNE problem maximizing the reliability and reducing the primary resource allocations as follows:

Variables:

- $F_{l^s}^{l^v}$: Binary variable denoting whether the substrate link l^s is used for the mapping of the virtual link l^v.
- $B_{bl^s}^{l^v}(l^s)$: Binary variable denoting whether the backup link bl^s protects against the failure risk l^s for the

flow l^v. In other words, $B^{l^v}_{bl^s}(l^s) = 1$ if and only if the flow of l^v will be rerouted on bl^s upon failure of link l^s.

- Y_{l^s} : Protection indicator defined as follows:

$$Y_{l^s} = \begin{cases} 1, & l^s \in L^S(VN) \text{ is protected} \\ 0, & \text{otherwise} \end{cases}$$

where $L^S(VN)$ corresponds to the set of primary substrate links used for *VN* embedding.

Objective:

Minimize:

$$\sum_{l^s \in L^S} [log(1-P_{l^s})] \times Y_{l^s} + \epsilon \sum_{l^s \in L^S} \frac{1}{\epsilon' + FR_{l^s}} \sum_{l^v \in L^V} F^{l^v}_{l^s}\, bw(l^v) \tag{1}$$

Subject to:

Primary flow constraints:

$$\sum_{\substack{n \in N^S | \\ \exists l^s = (m,n) \\ \in L^S}} F^{l^v}_{l^s} - \sum_{\substack{n \in N^S | \\ \exists l^s = (n,m) \\ \in L^S}} F^{l^v}_{l^s} = \begin{cases} 1, & m = M(src(l^v)) \\ -1, & m = M(dst(l^v)) \\ 0, & \text{otherwise} \end{cases} \quad \forall m \in N^s, l^v \in L^V \tag{2}$$

Primary bandwidth capacity constraints:

$$\sum_{l^v \in L^V} bw(l^v) F^{l^v}_{l^s} \leq FR_{l^s}, \qquad \forall l^s \in L^S \tag{3}$$

Backup flow constraints:

$$\sum_{\substack{n \in N^S | \\ \exists bl^s = (m,n) \\ \in L^S \setminus \{l^s\}}} B^{l^v}_{bl^s}(l^s) - \sum_{\substack{n \in N^S | \\ \exists bl^s = (n,m) \\ \in L^S \setminus \{l^s\}}} B^{l^v}_{bl^s}(l^s) = \begin{cases} F^{l^v}_{l^s} Y_{l^s}, & m = src(l^s) \\ -F^{l^v}_{l^s} Y_{l^s}, & m = dst(l^s) \\ 0, & \text{otherwise} \end{cases} \quad \forall m \in N^s, l^v \in L^V, l^s \in L^S \tag{4}$$

Backup bandwidth capacity constraints:

$$\sum_{l^v \in L^V} bw(l^v) B^{l^v}_{bl^s}(l^s) \leq BR_{bl^s}(l^s), \quad \forall bl^s \in L^S, l^s \in L^S \tag{5}$$

4.2 Explanation

Objective (1) is composed of two parts: (i) the failure probability and (ii) primary resource allocation. By multiplying the amount of primary resources by a very small constant ϵ ($0 < \epsilon \ll 1$), we ensure that the objective function (1) optimizes the reliability while reducing the primary resource allocations, i.e., objective (1) aims to minimize the primary bandwidth allocations (second part of the objective function) among all the solutions which maximize the reliability (first part of the objective function). More details about calculation leading to Objective (1) is given in Appendix.

For reliability optimization, the protectable links are preferred for primary virtual network embedding. When all the links are protectable ($\forall l^s \in L^S : \; Y_{l^s} = 1$), objective (1) ensures the determination of a VNE solution that minimizes the primary bandwidth. If some primary links cannot be protected due to the lack of resources ($\exists\, l^s \in L^S : \; Y_{l^s} = 0$), the survivability probability is maximized (the primary resources are minimized for only the most survivable solutions).

Equations (2) and (3) are primary flow and capacity constraints. $src(l^v)$ and $dst(l^v)$ retrieve respectively the source and destination node of l^v.

Equations (4) give backup flow constraints. In these equations, we assumed that a link can be fully protected against failures with the use of only one backup path (case of single failures). For ease of understanding, we deliberately keep the non linear version of these equations (the product of the binary variables $F^{l^v}_{l^s}$ and Y_{l^s} can be easily linearized). The equations (5) ensure that the amount of backup resources to be reserved on each link bl^s to protect against a given failure risk l^s are less than the available backup resources.

Note that the constraints (4) and (5) can be simplified by using a set of pre-computed backup paths (see Section 5.3.1). In this way, link protection is achieved with the selection of pre-computed backup paths minimizing the reliability. Even with such a backup path pre-computation, our VNE problem minimizing the reliability remains NP-hard because of primary and backup bandwidth constraints which require to respectively solve the multicommodity flow problem and the bin packing problem.

This ILP problem is a particular case of virtual network embedding, which is NP-hard. For scalability, we propose below heuristics to provide effective solutions in polynomial time.

Note that this formulation does not take into account other possible link attributes such as delay or financial cost. Indeed, adding such attributes leads to a multi-constrained/multi-objective problem which is NP-hard and beyond the scope of this paper.

5 HEURISTICS

Before proposing heuristics to solve the precedent ILP, we notice that:

- when the available primary bandwidth is higher than the virtual network demands, the constraints (3) are verified. In this case, the optimal solution correspond to a *Steiner Tree* maximizing the survivability probability (transformed to an additive metric in objective (1)). Among the well-known effective heuristics solving the Steiner problem, we cite the Kou-Markowsky-Berman's approach [29] where the terminal nodes are connected with use of shortest paths.
- when all the links are protectable, the left part (probability-based part) of objective (1) is nil. In this case, the problem formulated in precedent ILP becomes a classical multicommodity flow problem. Various heuristics exploring the k-shortest paths allows to determine efficient solutions.

From above, we propose two heuristics for survivable VNE problem. To provide high reliability, we re-defined the link costs according to objective function (1) and select the

protectable links before primary path computations. Both of our heuristics are based on Kou-Markowsky-Berman trees [29] which use shortest paths to connect the terminal nodes (substrate nodes embedding virtual nodes).

5.1 Principle

Reliability can be obtained by selecting the most reliable substrate links for VNE. With the single failure assumption, a protected substrate link is considered as reliable since it is always possible to recover from its failure. Thus, the primary substrate paths should privilege the use of protectable links to deal efficiently with failures.

Due to topological characteristics and insufficient resources, some virtual links can not be mapped to substrate paths which are composed only of protectable links. In this case, the substrate paths should be selected so that they minimize the failure probability of their unprotectable links.

Accordingly, we define below new positive primary link costs allowing the maximization of the reliability (by decreasing the probability of permanent VN failure):

$$cost(l^v, l^s) = \frac{\epsilon}{\epsilon' + FR_{l^s}} + \begin{cases} 0, & \text{if a backup path} \\ & \text{protecting } l^s \text{ exists} \\ -log(1 - P_{l^s}), & \\ & \text{otherwise} \end{cases} \quad (6)$$

As we see, the costs in (6) are composed of two parts: primary resource-dependent part that is multiplied by ϵ and failure-dependent part. The first part aims to decrease the primary resource allocations while the second part[2] ensures the reliability maximization. Indeed, for the protectable links, the primary costs depend only on resource allocations ($\frac{\epsilon}{\epsilon' + FR_{l^s}}$). As a result, the protectable links are selected so that they minimize the primary resource allocations. Concerning the unprotectable link costs, they depend mainly on failure probability ($-log(1 - P_{l^s})$). In this way, the cost minimization ensures the selection of the less vulnerable links. Among the minimal failure probability paths, the primary resource minimization criterion is used to select the best paths.

For each virtual link l^v, the primary cost $cost(l^v, l^s)$ depends on the protectability of substrate link l^s. Such cost can be computed on-line at the step of primary mapping or be more rapidly deduced by using pre-computed backup tunnels. Accordingly, we propose below two Kou-Markowsky-Berman tree-based heuristics, one for on-line computation (cf. 5.2), the other is the one based on pre-computation (cf. 5.3).

5.2 On-line backup verification

In our first proposition depicted in Algorithm 1, VNE is performed by mapping successively the virtual links. At each virtual link mapping, the primary link costs are computed on-line according to equation (6). In this step, the existence of a backup path is checked for each substrate link in order to deduce its cost. In this way, the virtual link is mapped on a substrate path that minimizes the costs in equation (6). Finally, links of the substrate primary path are protected by configuring backup paths verifying the bandwidth constraints.

2. To ensure positive costs, we used the objective function (10) (see Appendix) to define the failure-dependent parts in equation (6).

Algorithm 1 Link mapping with simple on-line backup verification

Input: virtual network request $G^V(N^V, L^V)$
Output: link mapping and backup solution
1: **for each** $l^v \in L^V$ **do**
2: **for each** $l^s \in L^S$ **do**
3: Set $cost(l^v, l^s) = \frac{\epsilon}{\epsilon' + FR_{l^s}}$;
4: Determine backup path π' for l^s, such that: $\forall bl^s \in \pi' : bl^s \neq l^s$ and $bw(l^v) \leq BR_{bl^s}(l^s)$;
5: **if** $\pi' = null$ **then**
6: $cost(l^v, l^s) = cost(l^v, l^s) - log(1 - P_{l^s})$
7: **end if**
8: **end for**
9: Find shortest $cost(l^v, l^s)$-based primary path π verifying the primary bandwidth capacity constraints;
10: **if** $\pi = null$ **then**
11: Free allocated resources;
12: **return** no solution;
13: **else**
14: Allocate primary resources for π;
15: **for each** $l^s \in \pi$ **do**
16: Determine backup path π' for l^s, such that: $\forall bl^s \in \pi' : bl^s \neq l^s$ and $bw(l^v) \leq BR_{bl^s}(l^s)$;
17: **if** π' $exists$ **then**
18: Allocate backup resources for π';
19: **end if**
20: **end for**
21: **end if**
22: **end for**
23: **return** link mapping and backup solution;

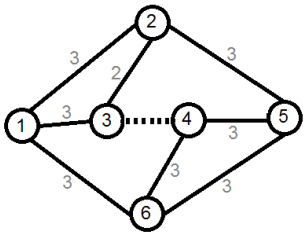
The details of our first proposition are shown in Algorithm 1. For each virtual link (line 1), the link costs are computed on-line (lines 2-8). Such link costs are determined by adding to the primary resource-dependent cost part (line 3) the failure-dependent cost part (line 6) that is nil for protectable links. Primary link mapping optimizing costs in equation (6) is determined in line 9. For each primary link, a local backup path is then determined in line 16.

When a primary mapping exists, primary and backup resources are allocated (lines 14 and 18) so that the next link mappings take them into account.

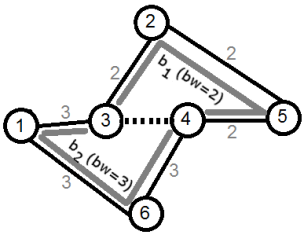
In Algorithm 1, we deliberately omit to specify the backup path search procedure (lines 4 and 16). By assuming a complexity of $\mathcal{O}(T)$ for the backup path search procedure, the complexity of Algorithm 1 is determined as equal to[3] $\mathcal{O}(|L^V|\ (|L^S|\ T + |N^S|\ \log_2 |N^S|))$. This means that the minimal worst-case time complexity corresponds to $\mathcal{O}(|L^V|\ |L^S|^2)$ since the quickest backup search procedure has complexity of[4] $\mathcal{O}(|L^S| + |N^S|) = \mathcal{O}(|L^S|)$. When the backup paths correspond to the shortest ones, the complexity is $\mathcal{O}(|L^V|\ |L^S|\ |N^S|\ \log_2 |N^S|)$.

3. Line 1 is performed in $\mathcal{O}(|L^V|)$. Lines 2 and 15 are performed in $\mathcal{O}(|L^S|)$. Line 9 is performed in $\mathcal{O}(|N^S|\ \log_2 |N^S|)$. Lines 4 and 16 are performed in $\mathcal{O}(T)$.

4. We assumed connected substrate graph where $|N^S| \leq |L^S| + 1$.



(a) Network



(b) Max-flow graph and backup tunnels maximizing the flows

Fig. 4: Maximum flow-based pre-determination of backup paths

5.3 Fast backup verification based on pre-computation

The on-line backup verification takes time $\mathcal{O}(|L^S|^2)$. To accelerate the computation and decrease the running time, we propose to pre-compute backup paths for each substrate link. Based on this pre-computation, the per request backup verification can be done for each substrate link in $\mathcal{O}(1)$, for a total complexity of $\mathcal{O}(L^S)$. The pre-computation needs to be done only when the substrate network topology changes. Hereafter, we describe the procedure of backup path pre-computation, then give algorithms decreasing the complexity of algorithm 1.

5.3.1 Pre-computation of backup paths

Due to the resource sharing enabled by the single failure assumption, it is possible to determine the maximum bandwidth that can be allocated to all backup paths protecting against a given link failure since such bandwidth does not depend on the other backup paths. In fact, the cumulated bandwidth of all backup paths which protect against the failure of a given link l^s is bounded by the maximum flow between the extremity nodes of l^s, after pruning link l^s from the network. Note that this upper bound is always reached when backup path flow splitting is enabled.

In Figure 4a, a network with 6 nodes and 9 links is depicted. All the links of the network have backup capacities equal to 3 units except link $2-3$ whose backup capacity corresponds to 2 units (see labels in Figure 4a).

To determine the backup paths with the highest cumulated bandwidth[5] (see algorithm 2), the maximum-flow graph should be determined offline for each substrate link (line 2 in algorithm 2) by using for instance Ford-Fulkerson's algorithm whose complexity is $\mathcal{O}(|N^S|\ |L^S|^2)$. Such a graph, which returns the amounts of flow on each link (link labels in Figure 4b), corresponds to an optimal solution to the maximum-flow problem. It allows us to determine the bypass tunnels (line 3 in algorithm 2) which maximize the flows. In the example depicted in Figure 4, two bypass tunnels $b_1 = (3-2-5-4)$ and $b_2 = (3-1-6-4)$, with respectively 2 and 3 units of bandwidth, are determined.

Assume that a first link mapping request with 2 units of bandwidth arrives. Link $3-4$, that is protectable (since there is a bypass b_2 with 2 available units of bandwidth), is used to satisfy the request. It is protected by a backup path that traverses the bypass tunnel b_2. Thus, only 3 units of bandwidth can be provided to protect against the failure of link $2-3$. Consider now that a second link mapping request with 3 units of bandwidth arrives. Link $3-4$ is protectable for this second request since there is at least one bypass tunnel b_1 with 3 available units of bandwidth. Assume that link $2-3$ is used and thus a second backup path traversing the bypass tunnel b_1 is setup. At this time, the link $3-4$ becomes unprotectable for any link mapping request since there is no free bandwidth on tunnels b_1 and b_2.

5.3.2 Algorithm

After solving offline the maximum flow problem for each substrate link, it is possible to quickly pre-determine for each link mapping request all the protectable links without path computation. Indeed, after the determination of the

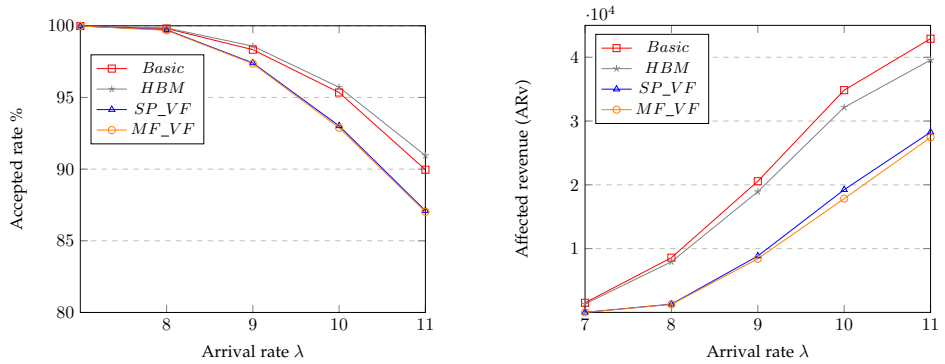
5. The upper bound to the cumulated bandwidth of backup paths is given by the maximum flow or the minimum cut. For the protected link $3-4$, the minimum cut is composed of links $1-3$ and $2-3$. Thus, the maximum flow between nodes 2 and 3 is equal to the sum of the capacities of these links, i.e. 5 units.

Algorithm 2 Max-flow based backup path pre-computation

Input: $G^S(N^S, L^S)$
1: **for each** $l^s \in L^S$ **do**
2: Determine the max-flow graph $MFGr(l^s)$ of link l^s on $G^S(N^S, L^S/l^s)$;
3: Determine a set of paths $\Pi(l^s)$ in $MFGr(l^s)$ such that the cumulated bandwidth of paths in $\Pi(l^s)$ corresponds to the max-flow value;
4: **end for**

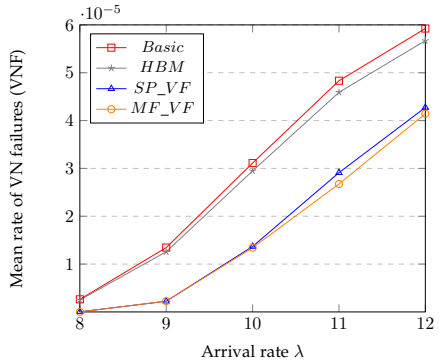
Algorithm 3 Link mapping with fast backup verification

Input: virtual network request $G^V(N^V, L^V)$
Output: link mapping and backup solution
1: **for each** $l^v \in L^V$ **do**
2: **for each** $l^s \in L^S$ **do**
3: Set $cost(l^v, l^s) = \frac{\epsilon}{\epsilon' + FR_{l^s}}$;
4: **if** $\forall \pi \in \Pi(l^s) : BR_\pi < bw(l^v)$ **then**
5: $cost(l^v, l^s) = cost(l^v, l^s) - log(1 - P_{l^s})$
6: **end if**
7: **end for**
8: Find shortest $cost(l^v, l^s)$-based primary path π verifying the primary bandwidth capacity constraints;
9: **if** $\pi = null$ **then**
10: Free allocated resources;
11: **return** no solution;
12: **else**
13: Allocate primary resources for π;
14: **for each** $l^s \in \pi$ **do**
15: Select one backup path $\pi' \in \Pi$ such that: $bw(l^v) \leq BR_\pi$
16: **if** π' $exists$ **then**
17: Allocate backup resources for π';
18: $BR_\pi \leftarrow BR_\pi - bw(l^v)$;
19: **end if**
20: **end for**
21: **end if**
22: **end for**
23: **return** link mapping and backup solution;

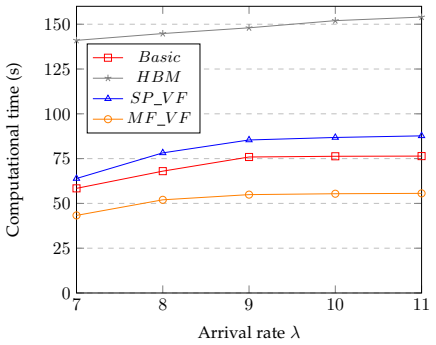


(a) Acceptance rate

(b) Affected revenue



(c) Mean rate of VN failures



(d) Computational time

Fig. 5: Cost239 network

link flows for each link failure, all the backup tunnels and their corresponding maximum bandwidth are deduced. When a link mapping request arrives, the unprotectable links are determined easily and quickly by searching for pre-determined backup tunnels verifying the bandwidth constraints (lines 4-6 in algorithm 3).

To summarize, it is clear that this second heuristic is based on a similar algorithm than that used by the first heuristic (see Section 5.2). The only difference between these heuristics consists in their backup path computation approaches (pre-computed backup tunnels for the second heuristic to reduce the computation time and on-line for the first one). More specifically, the fast heuristic runs Algorithm 3 upon reception of a VN request. With the use of the sets of pre-computed backup tunnels $\Pi(l^s)$, Algorithm 3 deduces quickly the costs of all the substrate links (lines 2-7) according to Equation (6). Recall that a substrate link l^s is protectable on a virtual link l^v if there is a backup path $\pi \in \Pi(l^s)$ such that $BR_\pi \geq bw(l^v)$.

After the cost determination step, a shortest primary path is determined (line 8). This path is then protected with the selection and configuration of pre-computed backup paths (lines 14-20).

To determine the worst-case time complexity of Algorithm 3, we remind you that the maximum number of tunnels in any set $\Pi(l^s)$ is lower than $|L^S|$. Indeed, after each backup tunnel setup, the available backup resources on one link at least becomes nil. With offline insertion of the backup tunnels in a priority queue, we deduce the worst-case time complexity of Algorithm 3 as $\mathcal{O}((|L^V|\ |L^S|\ \log_2|L^S|))$. Indeed, the backup path selection steps (lines 4 and 15 in Algorithm 3) are performed in constant times $\mathcal{O}(1)$ while updating the bandwidth allocation of a backup tunnel in the priority queue (lines 16 to 19 in Algorithm 3) takes $\mathcal{O}(\log_2|L^S|)$. Notice that this complexity does not take into account the pre-computation.

The reliability provided by Algorithm 3 can be improved by adopting a backup path preemption approach that requires more calculations. Indeed, at the step of cost determination of a given link, it is possible to recompute all the backup paths protecting that link for all virtual links already mapped. In this way, the bin packing problem is solved to optimize the costs and backup paths.

5.4 On-line vs. Fast backup verification

Both on-line and fast backup verification methods use similar on-line algorithms to compute the primary paths. The fast method ($\mathcal{O}(|L^V|\ |L^S|\ \log_2|L^S|)$) is quicker than the on-line method[6] ($\mathcal{O}(|L^V|\ |L^S|\ |N^S|\ \log_2|N^S|)$) once the backup tunnels pre-computation ($\mathcal{O}(|N^S|\ |L^S|^3)$) is accomplished. So, for a network with few substrate network topology changes, the fast one is more efficient. However, the pre-computation takes time and need to be done at each substrate network topology change. So, for a network with frequent substrate network topology changes, the on-line one could be an interesting alternative.

6. We consider the case where backup paths are the shortest ones.

6 SIMULATION

In order to measure the performance of our two proposals, we compared them against the basic method (protection method) and hybrid policy heuristic (restoration method) [7].

We implemented our proposals: (i) simple on-line backup verification-based heuristic (see Section 5.2) that is noted by SP_VF and (ii) fast backup verification-based heuristic (see Section 5.3) that is noted by MF_VF. In SP_VF, the backup paths are computed on-line as the shortest ones whereas we pre-computed the backup tunnels with MF_VF. In our simulations, we did not use preemption between the backup paths.

With the basic method (referred as $Basic$), we first determine the primary mapping that decreases the bandwidth allocation. For this purpose, the constrained shortest path algorithm is used. Then, each primary substrate path is protected with the use of local detours. With the hybrid policy method (referred as HBM), a set of k (k=6) shortest paths is pre-computed for each couple of substrate nodes. At the arrival of a new VN request, the multicommodity flow problem is solved to determine the best primary mapping. For survivability, a set of detours is also pre-computed. At the occurrence of a failure, the traffic traversing the failed link is rerouted on a set of detours that are determined and activated to deal with the failure. To reduce the total affected revenue (see Section 6.2) upon failure of link l^s, we chose to first repair networks G^V maximizing the ratio $\frac{Revenue(G^V)}{\sum_{l^v \in G^V} bw\,(l^v) \times F_{ls}^{lv}}$.

6.1 Environment

We developed a simulator to evaluate the performance of the compared methods. For our comparison study, we used two instances of real networks and a third one which is generated automatically with GT-ITM tool [30]. The first one (*Cost239*) is a small well connected network. The second one (*Germany50*) is a medium network that is less connected compared to the first one. The last one (*GT-ITM network*) is generated automatically. For each network, we set different substrate link failure probability models. Below, we only show the common configurations: the details of the model differences are described later in subsections that follow.

On the substrate networks, the node CPU and link bandwidth capacities are randomly chosen within (1000, 1500). For each substrate network, we fixed on links the ratio τ ($0.5 \leq \tau < 1$) between the primary bandwidth and the total capacity to a configurable value that depends on the network topology.

Concerning the virtual networks, they are generated by GT-ITM tool. The number of virtual nodes in each VN follows a uniform distribution between 3 and 8. The virtual nodes are interconnected with probability 0.4. The CPU and bandwidth demands are uniformly chosen in (0, 20).

In our experiments, we generated the requests according to Poisson distribution with different arrival rates λ per 100 time units. When a VN request arrives, we compute a valid mapping for all its virtual links. If such mapping is determined, the VN request is accepted. Then, the corresponding VN substrate links are protected with the use of backup paths.

To show the behavior and measure the performance of the compared methods for different network loads, we varied λ from 7 (low load) to 12 (high load).

The life time of each VN follows an exponential distribution with an average of 2000 time units. Each simulation lasts for 10^5 time units. The failure events of each substrate link follow an exponential distribution with a failure occurrence mean time of 10^4 units. All the substrate link failures are independent.

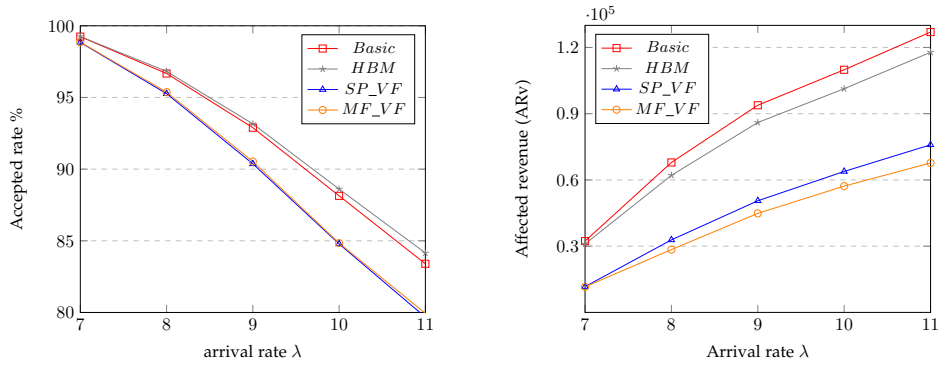
6.2 Comparison

We compared our proposals against methods determining the primary mapping without backup feasibility verification. In order to measure the performance of the compared methods, we used the following metrics:

- Acceptance rate of VNs (AR): It corresponds to the ratio between the accepted requests and the total number of requests. A VN request is accepted if all its corresponding virtual links are mapped on the substrate network. This metric is computed as follows:
$$AR = \frac{acc\_num}{tot\_num}$$
where acc_num denotes the number of accepted VNs and tot_num corresponds to the total number of VNs.
- Total affected revenue (ARv): When a failed virtual link is not repaired, the entire VN is assumed as failing. In this case, the service providers should pay a penalty for the failed VN. Such penalty is proportional to the VN revenue that is defined as follows:
$$\text{Revenue}\,(G^V) = a \sum_{n^v \in N^V} \text{cpu}\,(n^v) + \sum_{l^v \in L^V} \text{bw}\,(l^v)$$
This metric just accumulates the revenue of all the failed VNs (affected VNs). In our experiments, $a = 1$.
- Mean rate of VN failures (VNF): it corresponds to the ratio between the affected VNs and the mean number of active VNs in the network per time unit. Formally:
$$VNF = \frac{AF}{avr\_VN \times T}$$
where AF corresponds to the total number of affected VNs, avr_VN is the average of active VNs and T is the simulation time.
- Computational time (CT): this metric measures the time spent by the compared methods to perform all the computations. It provides insights about the time complexity.

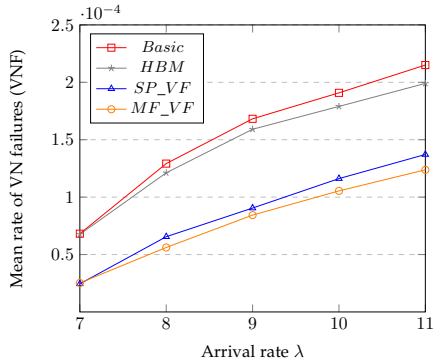
6.3 Scenario I: small size network

We first compared the methods on the European optical network (*cost239*) which contains 11 vertices and 26 edges. Cost239 is a well connected network with a minimum edge degree of 3 and a mean edge degree of 4.72. In this first scenario, we assumed an identical link failure model,

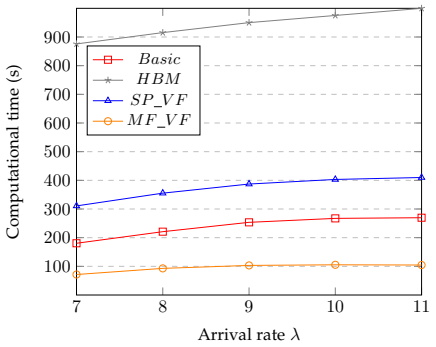


(a) Acceptance rate

(b) Affected revenue



(c) Mean rate of VN failures



(d) Computational time

Fig. 6: Germany50 network

i.e. all the link failures follow an exponential distribution with a mean time between failure occurrences equal to 10^4 time units. This model is useful for the networks where links have similar characteristics. The ratio τ between the primary bandwidth and the total link capacity corresponds to 0.75. We came up with this value through our simulations scenarios which provide high acceptance rates (larger than 99% with $\lambda = 7$).

The experiment results corresponding to this first scenario are shown in Figure 5. More specifically, Figure 5a, 5b, 5c and 5d depict respectively for different arrival rates λ the evolution of the acceptance rate of VNs, the total affected revenue, the mean rate of VN failures and the computational time. Note that all our results correspond to mean values over 100 experiments.

In Figure 5a, we see that HBM and $Basic$ have similar acceptances rates which are slightly better than those of our proposals. HBM and $Basic$ only focus on the primary mapping optimization so it is normal that these methods embed more VNs than our proposals. As expected, the backup feasibility verification has a side effect on the primary mapping that is slight here.

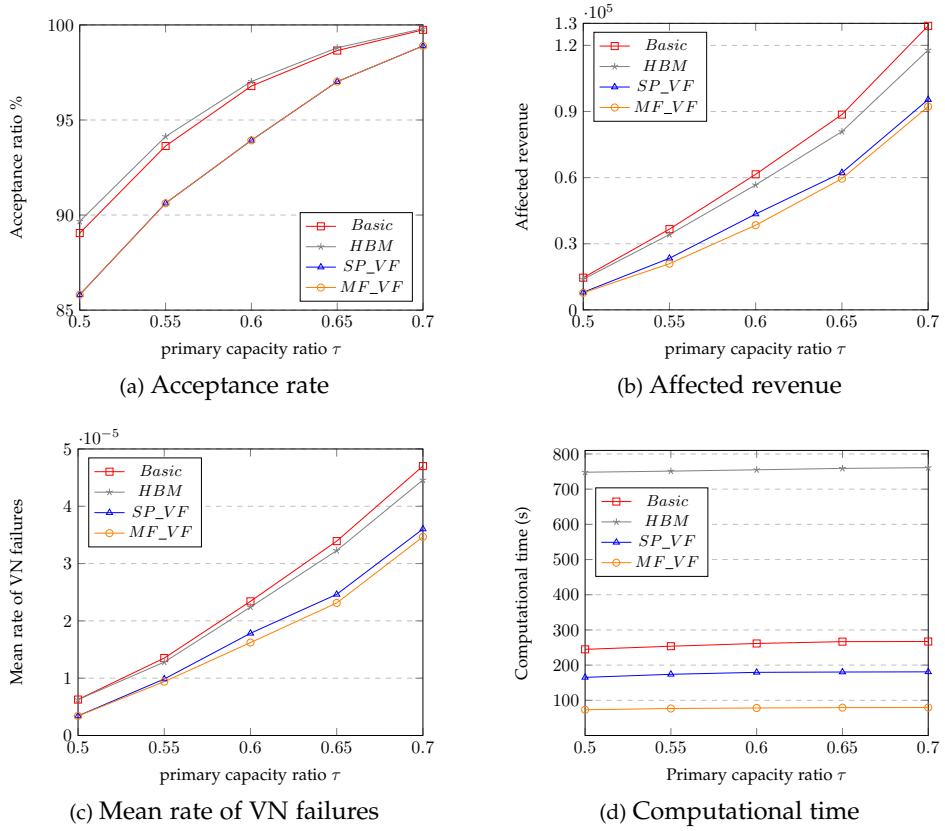
Concerning the survivability, Figures 5b and 5c show that our proposals clearly outperform HBM and $Basic$. Indeed, Figures 5b and 5c show that SP_VF and MF_VF have smaller affected revenues and VN failures than HBM and $Basic$. In addition, the difference between our proposals and the other ones increases with the augmentation of network loads (augmentation of λ). Whereas HBM and $Basic$ aim to minimize the primary resources without considering the backup resources, our proposals allow to significantly improve the reliability by choosing the more reliable links at the step of primary mapping. Thus, with slight decrease of the acceptance rate of primary VNs, the mean rate of VN failures is decreased by at least 30% as depicted in Figure 5c. We point out here that with the same level of reliability, our proposals are more efficient than HBM and $Basic$ in terms of resource utilization.

Note the very slight difference between SP_VF and MF_VF. This can be explained by the regularity and high connectivity of *Cost209*. In fact, the high flexibility in the selection of shortest backup paths (used by SP_VF) generally allows to maximize the flows (as with MF_VF).

With regards to the last metric (CT), Figure 5d shows that HBM spends a lot of time for doing the computations, compared to other methods. The figure also shows that MF_VF is better than $Basic$ which is in its turn more quick than SP_VF (and HBM). These results corroborate our theoretical results on time complexity. Whereas only the primary paths are computed on-line with MF_VF, $Basic$ also determines and computes the backup paths for all the substrate primary links. SP_VF is slower since it computes the backup paths for almost all substrate links.

6.4 Scenario II: medium size network

We retrieved a real medium network (*germany50*) from SNDlib. This network is composed of 50 vertices and 88 edges and it represents the network between German cities. The minimum edge degree in this network is 2 while the



(a) Acceptance rate (b) Affected revenue

(c) Mean rate of VN failures (d) Computational time

Fig. 7: Network topology generated by GT-ITM

mean edge degree is 3.52. The ratio of the primary bandwidth corresponds to 0.65 (this value allowed high acceptance rates in our simulations). *Germany50* is less connected than cost239 and there is nearly no direct link between extremity nodes. Therefore, the paths between extremity nodes are quite long.

Contrarily to scenario 1, the failure probabilities of links are different. They follow an exponential distribution with a mean time between failures (in time units) that is randomly chosen in the set $\{5 \times 10^3,\ 10^4,\ 1.5 \times 10^4\}$. This failure probability model is suitable for networks with different physical link characteristics.

Figures 6a, 6b and 6c show respectively the acceptance ratio, the total affected revenue and the mean rate of VN failures. Like in Figures 5a, 5b and 5c, we also observe that with slight decrease of acceptance ratio, our propositions often allow to divide the affected revenue and mean rate of VN failures by 2. In Figure 6d, we see that the computational time of MF_VF is smaller than those of $Basic$, SP_VF and HBM. With comparison to Figure 5d, we see that the difference between the compared methods is higher. Such observation can be explained by the increase of the primary path lengths which results in the computation of a larger number of backup paths.

To summarize, these observations allows us to conclude that with a slight diminution of VN acceptance, the backup feasibility verification significantly improves the reliability for different networks. Pre-computing backup paths by max-flow makes the backup feasibility verification method time saving.

6.5 Scenario 3: primary capacity ratio τ

In this scenario, we study the impact of primary capacity ratio τ on the performance of the compared methods. The substrate network, which is generated by GT-ITM, is composed of 30 nodes and 55 links. τ varies from 0.5 to 0.7. The VN arrival rate is 10. The link failure probability follows an exponential distribution with a mean time between failures (in time units) that is randomly chosen in the set $\{5 \times 10^3,\ 10^4,\ 1.5 \times 10^4\}$.

As expected, Figure 7a shows that the acceptance ratios of the compared methods increase with the augmentation of τ. The difference between our proposals and the methods $Basic$ and HBM is small and relatively stable, except for very high τ where our the acceptance rates of our methods decrease more quickly. When τ increases, the available backup bandwidth decreases and results in difficulty to protect the VNs. Whereas HBM and $Basic$ privilege the primary path optimization, our proposals may select longer primary paths to improve the protection.

Concerning the affected revenue and the mean rate of VN failures, Figure 7b and 7c show that our methods have close performance which are smaller than those of HBM and $Basic$. The figures also show that the difference increases with the augmentation of τ which results in the diminution of the backup resources.

With regards to the last metric (CT), Figure 7d shows that MF_VF is better than $Basic$ which is in its turn better than SP_VF and HBM. The difference between the compared methods can be explained in the same way as

in the precedent scenarios. However, in these last tests the computational time increases less quickly than in the precedent tests since the network load is constant ($\lambda = 10$ per 100 time units): only the acceptance rate of VNs increases with the augmentation of τ.

To conclude, this last scenario shows that the performance difference between our proposals and the methods HBM and $Basic$ increases with the diminution of the backup resources. Except for the time computation metric where MF_VF is always the best, there is almost no difference between the compared methods when the backup resources are very large.

6.6 Conclusion of simulation

The simulations allows us to conclude that:

- with a slight decrease of the acceptance rate of VNs, the backup feasibility verification significantly improves the reliability for different networks;
- pre-computing backup paths according to max-flow algorithm makes the backup feasibility verification method time-saving;
- with the same level of reliability, our backup feasibility verification methods are efficient in terms of resource utilization.

7 Conclusion

In a virtualized network environment, virtual networks share the same physical resources provided by substrate networks. A single link failure can therefore lead to dysfunction of several virtual networks. As a result, network reliability becomes a critical and, in our opinion, mandatory issue for virtual networks.

To enhance the VN reliability, survivable network design methods are used. Restoration changes the primary paths to accommodate the new network topology upon failure whereas protection pre-determines the backup paths before the failures so that the disruption time is reduced.

In order to guarantee enough resources upon failures, a portion of the resources should be pre-reserved for backup paths. Classical protection methods consider separately primary mapping and backup path computation, leading to non optimal resilience against failure.

In this paper, we proposed a novel virtual network protection framework, which maps the primary links in such a way that backup flow feasibility is taken into consideration. In our proposals, we first select the protectable links which reduce the resource allocations to support VNs. When full protection of VNs is not guaranteed, we proposed to include the less vulnerable links. After describing an exact solution that maximizes reliability while reducing resource allocations, we proposed two scalable, practical and efficient heuristics: simple on-line backup verification and fast backup verification. The first one determines on-line the backup paths whereas the second one pre-computes the backup paths to reduce the time computation.

Our proposals are validated by simulations. The numeric results show that our propositions reduce the VN failure probability with slight decrease of VN acceptance. The computational time can also be decreased by pre-computing the backup paths according to the maximum flow algorithm.

Appendix
Objective function

For a given VN request $G(N^V, L^V)$, a typical objective function optimizing the primary bandwidth corresponds:

$$min \quad \sum_{l^s \in L^S} \frac{1}{\epsilon' + FR_{l^s}} \sum_{l^v \in L^V} F_{l^s}^{l^v} \, bw(l^v) \tag{7}$$

where $F_{l^s}^{l^v}$ is a binary variable denoting that virtual link l^v is routed on a path including l^s, ϵ' ($0 < \epsilon' \ll 1$) is a small positive constant avoiding division by zero and FR_{l^s} corresponds to primary residual bandwidth on link l^s.
In this objective function, we chose to avoid flow splitting which is not so application-friendly on computer networks.

To maximize the reliability, objective (7) should be modified to include VN survivability probability $PS(Y)$ determined as follows:

$$PS(Y) = \prod_{l^s \in L^S} (1 - P_{l^s} \times (1 - Y_{l^s})) \tag{8}$$

where P_{l^s} corresponds to the failure probability of link l^s as described in Section 3.2.1 and Y_{l^s} is the protection indicator defined in Section 4.1.

To optimize the reliability, (8) should be maximized. Instead of maximizing the non linear function (8), we look for a linear function that is optimal for the same solution as (8). By applying logarithm function to (8), we obtain:

$$\begin{aligned} log\ PS(Y) &= \sum_{l^s \in L^S} log(1 - P_{l^s} \times (1 - Y_{l^s})) \\ &= \sum_{l^s \in L^S} [log(1 - P_{l^s})](1 - Y_{l^s}) \end{aligned} \tag{9}$$

(8) and (9) are maximized for the same solution. Instead of maximizing (9), we chose to minimize $-log\ PS(Y)$. Thus, the objective function that optimizes the reliability is given below:

$$min \sum_{l^s \in L^S} [-log(1 - P_{l^s})]\ (1 - Y_{l^s}) \tag{10}$$

By removing the constant part $\sum_{l^s \in L^S} -log(1 - P_{l^s})$ from objective function (10), we obtain the following simplified objective function:

$$min \sum_{l^s \in L^S} [log(1 - P_{l^s})] \times Y_{l^s} \tag{11}$$

Note that $log(1 - P_{l^s}) \times Y_{l^s}$ i minimal when $Y_{l^s} = 1$ since $log(1 - P_{l^s})$ is negative.

To decrease the primary bandwidth allocations while optimizing the reliability, we combined objectives (7) and (11). The resulting objective function is given by (1) in Section 4.1.

References

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *Computer*, no. 4, pp. 34–41, 2005.

[2] C. Colman-Meixner, C. Develder, M. Tornatore, and B. Mukherjee, "A survey on resiliency techniques in cloud computing infrastructures and applications," *Communications Surveys & Tutorials, IEEE*, vol. PP, no. 99, p. 1, 2016.

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

[4] S. Herker, A. Khan, and X. An, "Survey on survivable virtual network embedding problem and solutions," in *ICNS 2013, The Ninth International Conference on Networking and Services*, pp. 99–104, 2013.

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

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

[7] M. R. Rahman and R. Boutaba, "Svne: Survivable virtual network embedding algorithms for network virtualization.," *IEEE Transactions on Network and Service Management*, vol. 10, no. 2, pp. 105–118, 2013.

[8] S. Li, M. Y. Saidi, and K. Chen, "A Failure Avoidance oriented Approach for Virtual Network Reliability Enhancement," in *IEEE International Conference on Communications, ICC 2017, Paris, France, May 21-25, 2017*, pp. 1–6, 2017.

[9] A. Belbekkouche, M. Hasan, and A. Karmouch, "Resource discovery and allocation in network virtualization," *Communications Surveys & Tutorials, IEEE*, vol. 14, no. 4, pp. 1114–1128, 2012.

[10] J. Zhang, H. Huang, and X. Wang, "Resource provision algorithms in cloud computing: A survey," *J. Network and Computer Applications*, vol. 64, pp. 23–42, 2016.

[11] D. Xu, Y. Chen, Y. Xiong, C. Qiao, and X. He, "On the complexity of and algorithms for finding the shortest path with a disjoint counterpart," *IEEE/ACM Transactions on Networking*, vol. 14, no. 1, pp. 147–158, 2006.

[12] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, 2008.

[13] I. Houidi, W. Louati, W. B. Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Computer Networks*, vol. 55, no. 4, pp. 1011–1023, 2011.

[14] Y. Zhu and M. H. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *INFOCOM*, vol. 1200, pp. 1–12, 2006.

[15] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pp. 81–88, ACM, 2009.

[16] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 38–47, 2011.

[17] N. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *INFOCOM*, pp. 783–791, IEEE, 2009.

[18] A. Khan, X. An, and S. Iwashina, "Virtual Network Embedding for telco-grade network protection and service availability," *Computer Communications*, vol. 84, pp. 25–38, 2016.

[19] H. Jiang, L. Gong, and Z. W. Zuqing, "Efficient joint approaches for location-constrained survivable virtual network embedding," in *IEEE Global Communications Conference, GLOBECOM 2014, Austin, TX, USA, December 8-12, 2014*, pp. 1810–1815, 2014.

[20] W.-L. Yeow, C. Westphal, and U. C. Kozat, "Designing and embedding reliable virtual infrastructures," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 57–64, 2011.

[21] H. Yu, V. Anand, C. Qiao, and G. Sun, "Cost efficient design of survivable virtual infrastructure to recover from facility node failures," in *Communications (ICC), 2011 IEEE International Conference on*, pp. 1–6, IEEE, 2011.

[22] H. Yu, V. Anand, and C. Qiao, "Virtual infrastructure design for surviving physical link failures," *The Computer Journal*, 2012.

[23] T. Guo, N. Wang, K. Moessner, and R. Tafazolli, "Shared backup network provision for virtual network embedding," in *Proc. IEEE Int. Conf. Communications (ICC)*, pp. 1–5, June 2011.

[24] Z. Wang, J. Wu, Y. Wang, N. Qi, and J. Lan, "Survivable virtual network mapping using optimal backup topology in virtualized sdn," *China Communications*, vol. 11, pp. 26–37, Feb. 2014.

[25] A. Jarray and A. Karmouch, "Cost-efficient mapping for fault-tolerant virtual networks," *IEEE Transactions on Computers*, vol. 64, pp. 668–681, Mar. 2015.

[26] H. W. Lee, E. Modiano, and K. Lee, "Diverse routing in networks with probabilistic failures," *IEEE/ACM Transactions on Networking*, vol. 18, pp. 1895–1907, Dec. 2010.

[27] R. L. Gomes, L. F. Bittencourt, E. R. Madeira, E. Cerqueira, and M. Gerla, "Bandwidth-aware allocation of resilient virtual software defined networks," *Computer Networks*, vol. 100, pp. 179–194, 2016.

[28] H. Yang, L. Cheng, G. Luo, J. Zhang, Y. Zhao, H. Ding, J. Zhou, and Y. Wang, "Survivable virtual optical network embedding with probabilistic network-element failures in elastic optical networks," *Optical Fiber Technology*, vol. 23, pp. 90–94, 2015.

[29] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for steiner trees," *Acta informatica*, vol. 15, no. 2, pp. 141–145, 1981.

[30] GT-ITM *http://www.cc.gatech.edu/projects/gtitm/*.