國立臺灣大學電機資訊學院電信工程學研究所

碩士論文

Graduate Institute of Communication Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master's Thesis

針對單一實體設施節點失效之具生存性虛擬網路可擴展映射方法

Scalable Embedding of Survivable Virtual Network Against Single Facility Node Failures

呂霽原

JI-YUAN LU

指導教授：周俊廷 博士

Advisor: Chun-Ting Chou, Ph.D.

中華民國 114 年 4 月

April 2025

doi:10.6342/NTU202501591

致謝

在台大的這幾年過著悠閒的生活，直到去年三月底時老師提醒我後才驚覺不行再這樣放鬆下去，於是經過了幾個月跟老師不斷討論和尋找題目，在八月中時找到題目，並經過了幾個月完成實驗後，最後在十一月初開始把研究結果寫成論文。過程中由衷感謝周俊廷老師教我如何挑選值得投入的題目，報告時如何能夠清楚地表達重點，以及撰寫論文時的各種技巧跟眉角。相信這些過程中累積的思維和邏輯能力，在未來不論做什麼事情都會是很重要的基礎。也感謝擔任口試委員的魏宏宇教授、逄愛君教授、蕭旭君教授指出論文中的不足，讓我能夠進一步修正與補強。

感謝這段期間的實驗室同學丞傑、紹銘在研究過程的討論，常常指出思考上的盲點，讓我能從正確的角度解決問題。還有謝謝系辦的趙文瑛小姐在心靈上的開導，也謝謝 Anita 助理在老師借調期間協助處理行政上的聯繫與事情。最後謝謝我的家人提供環境還有經濟上的幫助，讓我能夠專注在研究上，才能順利完成碩士論文。

doi:10.6342/NTU202501591

中文摘要

隨著網路基礎設施對於有彈性且高效資源利用的需求日益增長，虛擬網路嵌入（VNE）技術應運而生。虛擬網路（VN）已廣泛應用於雲端計算與軟體定義網路（SDN）等應用中，以滿足這些不同應用或服務需要具有獨立的網路拓撲結構、特殊計算資源、網路頻寬、延遲與可靠性之多樣需求。透過 VN 的需求抽象化，網路營運商在無需關注底層實體網路的細節下仍然能夠提供相應之服務，同時透過虛擬網路嵌入（VNE）演算法，營運商可快速將 VN 對應至實體網路（SN），確保服務品質（QoS）需求得以滿足，同時最大化資源利用率。

在 VNE 研究領域的眾多議題中，本研究聚焦於因實體網路（SN）的失效（尤其是節點失效）時產生的生存性問題，實體網路之失效可能源自於硬體故障、軟體錯誤或網路攻擊，對於關鍵應用與服務而言，此類失效可能嚴重影響使用者滿意度與營運可靠性，因此可生存性成為 VN 設計與管理中不可忽視的議題。

為解決 VNE 的可生存性問題，本研究採用一種解耦方法，也就是將生存性與嵌入問題分開處理 [16][17]，並提出兩項新的設計原則。首先，我們在所謂的增強階段（augmentation phase）處理生存性問題，透過增加備援虛擬節點與相應的備援虛擬連結來「增強」 VN。此研究透過探討各種增強 VN 的策略，識別出一種擁有最大 augmented VN 解空間的增強策略，提高了尋找全域最佳 augmented VN 的可能性。其次，本研究開發了一種名為 Nest-Base 的演算法，以快速識別在放寬限制條件下的全域最佳 augmented VN。此外，本研究亦設計了一種嵌入演算法，此演算法確保了增強階段的資源增量與最終嵌入成本之間具有較強的線性關聯性，使得 Nest-Base 找到的資源增量最小的

doi:10.6342/NTU202501591

augmented VN，能夠最終實現最低的嵌入成本。

模擬結果顯示，Nest-Base 在 VN 接受率（acceptance ratio）與嵌入成本方面皆優於現有方法，包括 FIP、FDP [16]及 ProRed [17]。在適中負載的網路環境下，Nest-Base 嵌入成本降低幅度 在 小型 SN 中相較於 FIP 降低 11.6% 至 29.2%，相較於 FDP 降低 12.9% 至 31.5%，相較於 ProRed 降低 18.9% 至 25.4%。在 大型 SN 中，嵌入成本的降低幅度分別為 10.0% 至 29.1%（相較於 FIP）、11.7% 至 31.0%（相較於 FDP）、19.5% 至 25.7%（相較於 ProRed）。

在高負載的網路環境下，Nest-Base VN 接受率範圍 在 小型 SN 中為 60% 至 70%，在 大型 SN 中為 85% 至 100%。相較於 FIP、FDP 和 ProRed，Nest-Base 在 小型 SN 的 VN 接受率分別提升 25% 至 35%（相較於 FIP）、30%（相較於 FDP）、10% 至 15%（相較於 ProRed）。在 大型 SN 中，VN 接受率的提升幅度為 25% 至 45%（相較於 FIP 和 FDP）、2% 至 12.5%（相較於 ProRed）。此結果凸顯了本研究方案的實用性，並顯示其在多種實際網路應用場景中的價值。

關鍵詞： 虛擬網路嵌入（VNE）、可生存性 VNE、VN 增強、節點失效

doi:10.6342/NTU202501591

ABSTRACT

The growing demand for flexible and efficient resource utilization of network infrastructures has driven the development of Virtual Network Embedding (VNE). Virtual Networks (VNs) are widely used in cloud computing and software-defined networks (SDN), where different applications or services require independent, isolated network topologies with specific requirements in terms of bandwidth, latency, and reliability. The abstraction of VNs allows network operators to provide these services without worrying about the underlying physical network's details. By using VNE algorithms, the VN can be efficiently mapped onto the substrate networks (SNs), ensuring that the QoS requirements are satisfied, and resource utilization is maximized.

Among different issues studied in the field of VNE, we focus on the survivability issue which deal with SNs failure (in particular node failure) due to hardware malfunctions, software bugs, or cyberattacks. For critical applications and services, these failures can severely affect user satisfaction and operational reliability, making survivability an essential consideration in VN design and management.

To address the survivability issue of VNE, we adopted a decoupled approach to tackle survivability and embedding separately [16][17] with two new design principles. First, the survivability issue is handled in the so-called augmentation phase, where backup virtual nodes are added, along with corresponding backup virtual links, to "augment" the VNs. By exploring various augmented VN strategies, this research

doi:10.6342/NTU202501591

identifies an augmentation strategy with the largest solution space for augmented VNs, thereby enhancing the likelihood of finding a optimal augmented VN. Second, an algorithm named Nest-Base was developed to quickly identify optimal augmented VNs under the relaxed constraints. Finally, a mapping (embedding) algorithm with a stronger linear correlation between resource increment of the augmentation phase and embedding cost of the mapping (embedding) phase is also developed such that the augmented VN with the minimum resource increment during the augmentation phase will lead to the minimum mapping (embedding) cost during the mapping phase.

The simulation results indicate that Nest-Base provides the lowest embedding cost while accommodating VNs significantly better than other approaches, including FIP, FDP [16], and ProRed [17]. Under moderate-load network conditions, Nest-Base achieves an embedding cost reduction ranging from 11.6% to 29.2% compared to FIP, 12.9% to 31.5% compared to FDP, and 18.9% to 25.4% compared to ProRed in small-scale SNs. In large-scale SNs, the embedding cost reduction reaches 10.0% to 29.1% over FIP, 11.7% to 31.0% over FDP, and 19.5% to 25.7% over ProRed.

Under high-load network conditions, the VN acceptance ratio of Nest-Base remains within 60% to 70% in small-scale SNs and 85% to 100% in large-scale SNs. Compared to FIP, FDP, and ProRed, Nest-Base achieves a VN acceptance ratio improvement ranging from 25% to 35% over FIP, 30% over FDP, and 10% to 15% over ProRed in small-scale SNs. In large-scale SNs, the VN acceptance ratio improvement reaches 25% to 45% over FIP and FDP, and 2% to 12.5% over ProRed.

This highlights the practicality of our solution, underlining its value across a variety of potential networking scenarios.

doi:10.6342/NTU202501591

Keywords: virtual network embedding (VNE), survivability VNE, VN augmentation, node failure

doi:10.6342/NTU202501591

CONTENTS

doi:10.6342/NTU202501591

doi:10.6342/NTU202501591

doi:10.6342/NTU202501591

doi:10.6342/NTU202501591

LIST OF FIGURES

doi:10.6342/NTU202501591

doi:10.6342/NTU202501591

doi:10.6342/NTU202501591

doi:10.6342/NTU202501591

doi:10.6342/NTU202501591

doi:10.6342/NTU202501591

LIST OF TABLES

doi:10.6342/NTU202501591

doi:10.6342/NTU202501591

doi:10.6342/NTU202501591

LIST OF ALGORITHMS

doi:10.6342/NTU202501591

CHAPTER 1
INTRODUCTION

1.1 Background of Virtual Network Embedding

The growing demand for flexible, scalable, and efficient resource utilization in network infrastructures [1] has driven the development of Virtual Network Embedding (VNE). VNE addresses the need to efficiently map virtual networks (VNs) [2], which represent customizable and isolated network topologies, onto a shared substrate network (SN).

A Virtual Network (VN) is a logical, abstracted representation of a network that is decoupled from the physical substrate network. It consists of virtual nodes and virtual links that represent computing resources (such as virtual machines or storage) and communication pathways (such as virtualized network connections), respectively [3]. VNs are typically used in cloud computing, telecommunications, and software-defined networking (SDN) environments, where different applications or services require independent, isolated network topologies with specific requirements in terms of bandwidth, latency, and reliability.

For example, consider an Infrastructure as a Service (IaaS) provider offering cloud services [4]. The provider may host several virtual private networks (VPNs) for different customers, each requiring different configurations based on their needs. One customer might need a high-throughput network with minimal latency for video streaming services, while another might need a more secure, low-bandwidth network

doi:10.6342/NTU202501591

for financial applications. These service requirements can be abstracted into VNs, each with its own set of virtual nodes (representing virtual machines) and virtual links (representing virtualized communication paths), mapped to the physical resources of the substrate network.

The SN is the physical network infrastructure that provides the resources upon which virtual networks are mapped. It consists of substrate nodes (the physical devices, such as servers, routers, or switches) and substrate links (the physical communication paths connecting these nodes, such as optical fibers or Ethernet cables). The substrate network serves as the foundation for hosting multiple virtual networks, and resources from the substrate are shared among different virtual networks.

The abstraction of VNs allows network operators to provide these services without worrying about the underlying physical network's details. By using VNE algorithms, the VN can be efficiently mapped onto the physical infrastructure, ensuring that the QoS requirements are met and resource utilization is maximized [3]. This enables flexible network management and the co-hosting of multiple, diverse VNs on the same physical infrastructure, ensuring optimal use of the substrate network.

1.2 Background of Survivable Virtual Network Embedding

In SNs, node or link failures can occur due to various reasons such as hardware malfunctions, software bugs, natural disasters, or cyberattacks. These failures can disrupt the VNs mapped onto the physical infrastructure, causing loss of connectivity or functionality.

doi:10.6342/NTU202501591

Such disruptions in VNs may lead to significant consequences, including service outages, degraded performance, and potential financial losses. For critical applications and services, these impacts can severely affect user satisfaction and operational reliability, making survivability an essential consideration in VN design and management.

Survivable Virtual Network Embedding (SVNE) is a research domain aimed at addressing this issue [5]. Unlike VNE, which primarily focuses on the efficient mapping of virtual networks onto substrate networks, SVNE explicitly incorporates survivability considerations into the embedding process. It focuses on embedding a VN onto an SN in a way that ensures the VN can continue to function normally, even when part of the SN experiences failures.

The primary goal of SVNE is to solve the VNE problem under the constraint of guaranteed survivability. This involves not only mapping virtual nodes and links to physical resources efficiently but also ensuring that the VN can maintain its functionality and service requirements despite potential failures in the substrate network. These failures could include node failures, link failures, or even the failure of a combination of nodes and links [6].

One of the key challenges in SVNE is its inherent complexity, which makes it an NP-hard problem. This complexity stems from the fact that VNE, a foundational problem for SVNE, is itself NP-hard. The NP-hard nature of VNE arises because it involves finding an optimal mapping that minimizes embedding costs while satisfying the resource constraints of both the virtual and substrate networks [7]. SVNE, building on this complexity, introduces an additional challenge by requiring the embedding to

doi:10.6342/NTU202501591

ensure survivability. This means that SVNE not only needs to address the optimization and constraint satisfaction issues of VNE but also must guarantee that the virtual network remains functional despite substrate failures. These additional survivability requirements further intensify the complexity of the problem, making SVNE even more challenging to solve.

Due to the NP-hard nature of SVNE, many existing research papers simplify the problem by focusing on either link failures or node failures, rather than addressing both types of failures simultaneously. Among these, studies dealing with link failures are more prevalent than those addressing node failures [8]. This discrepancy arises because link failures tend to occur more frequently than node failures in substrate networks [9]. Consequently, ensuring survivability against link failures has become a primary focus in SVNE research.

1.3 Motivation

In network failure scenarios, link failures are significantly more common than node failures. As a result, existing research has predominantly focused on addressing link failures. However, node failures can similarly have a substantial impact on the stability of network services, underscoring the importance of developing strategies to address node failures effectively.

In this study, we concentrate on tackling the problem of single facility node failures, which represents the most fundamental form of node survivability challenges. This foundational research serves as a basis for addressing more complex node survivability problems.

doi:10.6342/NTU202501591

VNE primarily focuses on solving the mapping problem, i.e., efficiently embedding virtual nodes and links onto the substrate network. In contrast, SVNE in the context of single facility node failures not only requires solving the mapping problem but also incorporates survivability considerations. How to efficiently address both survivability and mapping in single facility node failures scenarios remains a challenging task.

Studies such as [10], [11], and [12] propose solutions that directly address the SVNE problem by embedding virtual networks onto the SN while simultaneously ensuring survivability. However, the large scale of SN makes it computationally infeasible to efficiently solve such an NP-hard problem directly on the substrate network. This inherent complexity limits the practical applicability of these methods, as they struggle to balance efficiency and optimality in real-world scenarios.

To address these challenges, some studies [13]-[17] have proposed a decoupled approach to tackle survivability and mapping separately. Specifically, these approaches first augment the VN by integrating survivability requirements into the VN, which results in an augmented VN with additional backup resources. Then, the augmented VN is mapped onto the substrate network during the mapping (embedding) phase, focusing solely on solving the mapping problem.

This decoupled strategy offers the following advantages:

1. **Problem Size Reduction:**

   As the size of VN is generally smaller than that of the substrate network, solving the survivability problem at the VN level significantly reduces

doi:10.6342/NTU202501591

problem complexity.

2. **Increased Flexibility in the Mapping (Embedding) Phase:**
   Since the survivability requirements are already addressed during the VN augmentation phase, the mapping (embedding) phase can focus on optimizing a single objective, such as resource utilization efficiency or VN acceptance rate. This allows for specialized algorithms to achieve lower mapping (embedding) costs and higher efficiency in resource usage.

Although previous studies [13]-[15] proposed methods for constructing augmented VNs by increasing backup resources to enhance survivability, they did not analyze how the amount of backup resources impacts the mapping (embedding) cost when the VN is mapped onto the SN. As a result, although their approaches aimed to improve survivability at the VN level, they did not lead to cost-effective solutions for the overall SVNE problem when embedded in the SN.

In contrast, studies in [16] and [17] not only introduced backup resources in the augmented VN to enhance survivability but also considered the impact of these backup resources on the mapping (embedding) cost. They assume that:

"**The augmented VN with a smaller resource increment during the augmentation phase will lead to a lower mapping (embedding) cost during the mapping (embedding) phase, and the augmented VN with the smallest resource increment can achieve the globally optimal mapping (embedding) cost**."

Guided by this assumption, their methodologies focus on minimizing the resource increment required for survivability during the VN augmentation phase, aiming to

doi:10.6342/NTU202501591

achieve the lowest mapping (embedding) cost in the subsequent mapping (embedding) phase.

However, upon analyzing these methods, we identify significant limitations in their design, which invalidate the aforementioned assumption:

1. **Lack of a Linear Relationship Between the Increment in the Augmented VN and Mapping (Embedding) Cost:**
   In practice, augmented VN with less increment tend to result in lower mapping (embedding) costs. However, this correlation is not guaranteed. In some cases, augmented VN with minimal increment may lead to higher mapping (embedding) costs, while those with larger increment may achieve lower mapping (embedding) costs.

2. **Overly Narrow Solution Space Caused by Augmentation Constraints:**
   Excessive constraints imposed during the VN augmentation phase result in a highly narrow solution space for possible augmented VN. A smaller solution space reduces the likelihood of containing the globally optimal solution—i.e., the augmented VN that leads to the minimum mapping (embedding) cost.

The insufficient linearity between the resource increment of the augmented VN and the mapping (embedding) cost indicates that mapping algorithms do not effectively account for the influence of resource increments on embedding outcomes. Furthermore, excessive constraints during VN augmentation limit the solution space. To address these issues, it is necessary to redesign mapping algorithm to better linearity between resource increment and mapping (embedding) cost, and adopt less

doi:10.6342/NTU202501591

restrictive augmentation method to expand the solution space, enhancing the chances of achieving globally optimal embeddings.

1.4 Research Objectives and Problem Definition

The challenges outlined in Section 1.3 emphasize the need for innovative approaches to address SVNE under single facility node failures. Existing methods [16],[17] for generating augmented VNs and embedding them onto SNs exhibit significant limitations.

This thesis addresses these issues by

1. **Designing a Mapping Algorithm to Ensure Linearity Between Resource Increment and Mapping (Embedding) Cost:**

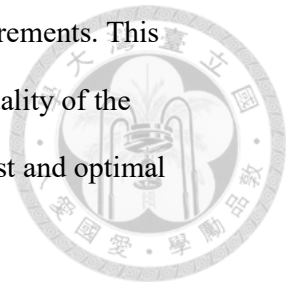
   The lack of a linear relationship between resource increments and mapping (embedding) costs is a major issue discussed. Therefore, one of the primary objectives of this research is to design a mapping algorithm that addresses this issue by ensuring a linear relationship between the resource increments in augmented VN and the resulting mapping (embedding) cost.

2. **Design for Augmented VN Generation Algorithm with Relaxed Constraints:**

   A key issue identified in the motivation is the overly restrictive constraints in the current augmented VN generation strategies. This research aims to design a more flexible process for generating augmented VN with fewer constraints, thus expanding the solution space and increasing the chances of finding globally optimal solutions. An algorithm approach will be developed to enable

doi:10.6342/NTU202501591

the rapid generation of augmented VN with minimal resource increments. This algorithm will prioritize both computational efficiency and the quality of the generated augmented VN, aiming to produce solutions that are fast and optimal in terms of resource increments.



3. **Validating the Performance of the Proposed Algorithms under the Relaxed Constraints**

   This study aims to evaluate the proposed mapping algorithm and validate the predictable linear relationship between resource increments and mapping (embedding) costs. With the help of linearity, we can ensure that the minimally incremented augmented VN, when mapped using the proposed algorithm, achieves the lowest mapping (embedding) cost. Second, to demonstrate that the lowest mapping (embedding) cost achieved through this approach can be found using the proposed augmentation algorithm and is expected to be smaller than the minimal mapping (embedding) cost achievable under existing methods [16],[17].

In summary, this research focuses on scenarios involving a single facility node failure in the context of the SVNE problem. Given a substrate network (SN) and a virtual network request (VN request), the objective is to determine an augmented VN that ensures survivability while minimizing the mapping (embedding) cost.

1.5 Thesis Contributions

This thesis makes several key contributions to the field of SVNE, particularly in the design of augmented VN strategies and design of augmented VN algorithm and

doi:10.6342/NTU202501591

mapping algorithms that ensure minimal embedding costs. The main contributions are summarized as follows:

1. We introduce an augmented VN strategy, MUPL, which has the highest probability of containing the globally optimal augmented VN. By comparing the solution space sizes of various augmented VN strategies, we confirm that MUPL's solution space is sufficiently large to almost always include the globally optimal augmented VN in small-scale VNs.

2. To ensure a predictable relationship between resource increment and embedding cost, we propose a mapping algorithm with linear properties, guaranteeing a linear correlation between an augmented VN's resource increment and its embedding cost.

3. To efficiently identify the globally optimal augmented VN within the MUPL strategy's solution space, we develop the Nest-Base algorithm, which significantly reduces computational complexity while maintaining optimality. By integrating the Nest-Base algorithm with the proposed linear mapping algorithm, we successfully achieve globally optimal embedding costs in small-scale scenarios.

4. We conduct a numerical analysis using exhaustive search on small-scale VNs, demonstrating that the Nest-Base algorithm, when combined with the linear mapping algorithm, consistently finds augmented VNs with lower minimal embedding costs than existing methods, ultimately reaching the globally minimum embedding cost.

doi:10.6342/NTU202501591

5. Finally, we simulate real-world network conditions by designing scenarios with both moderate and high substrate network loads. Under moderate loads, our approach reduces embedding costs by at least 12% compared to existing methods. Under high-load conditions, our approach improves the VNR acceptance rate by at least 10%, demonstrating its adaptability and robustness in resource-constrained environments.

1.6 Thesis Organization

This thesis is organized into six chapters as follows. Chapter 2 reviews the related work on VNE and SVNE. Chapter 3 describes the system settings and assumptions, including network models, resource constraints, and evaluation metrics. Chapter 4 presents the proposed methods, including the Nest-Base algorithm and a mapping algorithm with linear properties. Chapter 5 validates the proposed methods through numerical analysis using exhaustive search and simulations that model realistic scenarios, and compares their performance with existing approaches. Finally, Chapter 6 concludes the thesis and discusses future research directions.

doi:10.6342/NTU202501591

CHAPTER 2
RELATED WORK

This chapter provides a review of related work on virtual network embedding (VNE) and survivable virtual network embedding (SVNE). Section 2.1 focuses on existing studies addressing the key challenges of VNE, particularly the issues related to resource allocation and the embedding of virtual networks onto a substrate network (SN) that is assumed to be failure-free.

Section 2.2 transitions to the domain of SVNE, which explores the survivability of virtual networks under substrate failure. The discussion is divided into two subsections: Section 2.2.1 focuses on addressing substrate link failure, while Section 2.2.2 focuses on substrate node failure.

Within Section 2.2.2, the discussion is further subdivided into two subsections: Section 2.2.2.1 and Section 2.2.2.2. The former investigates scenarios where a failed substrate node causes the failure of its adjacent substrate links, whereas the latter considers cases where the adjacent substrate links remain functional despite the node failure.

Section 2.3 identifies the limitations of existing augmented VN approaches. Section 2.3.1 examines the assumptions in [16] and [17] regarding the relationship between resource increment and mapping cost. Section 2.3.2 focuses on the narrow solution space of [17] and its impact on how low the minimum mapping cost can be.

doi:10.6342/NTU202501591

2.1 Virtual Network Embedding

VNE research has explored various challenges in SN resource management and allocation. Both [18] and [19] focus on addressing network overload issues in SN. [18] primarily addresses overloaded substrate links, which lead to virtual network request (VNR) rejections, while [19] focuses on server overloads in substrate nodes, specifically dealing with server crashes caused by excessive demand.

In addition to network overload, resource allocation inefficiencies within the VNE process have also been a key area of focus. Both [20] and [21] address these inefficiencies, albeit from different perspectives. [20] resolves the lack of coordination between node and link mapping in the traditional two-phase VNE process, while [21] explores the relative importance of substrate nodes by using a node ranking method to improve resource allocation efficiency.

Expanding on these resource challenges, [22] and [23] investigate the VNE problem in fog computing networks, where substrate nodes differ in computational capabilities, functionality, and operational costs. [22] focuses on environments with heterogeneous functional substrate nodes, while [23] addresses the challenge of solving VNE under varying per-unit resource costs of substrate nodes while meeting low-latency constraints.

Finally, [24] highlights the complexity of embedding VN across multiple SNs managed by different infrastructure providers (InPs). This complexity arises from the varying policies of each SN and the competition among InPs to maximize the utilization of their high-profit resources.

doi:10.6342/NTU202501591

2.2 Survivable Virtual Network Embedding

VNE studies have explored resource optimization through techniques such as load balancing, coordinated node and link mapping [18]-[24]. However, these studies are often based on simplified assumptions, such as stable substrate networks without node or link failures, limiting their applicability in real-world scenarios.

To address these limitations, SVNE expands the scope of VNE by incorporating backup resources and recovery mechanisms to manage potential failures. Researchers commonly divide the SVNE problem into two subproblems: addressing link failures and addressing node failures. This decomposition allows for more focused discussions on the unique challenges posed by each failure scenario. The following sections discuss the challenges associated with these subproblems.

2.2.1 Link Failure in Virtual Network Embedding

In [25], the authors proposed an approach to ensure the survivability of virtual networks under single substrate link failures by allocating a disjoint backup path for each primary path mapped during the embedding process. In [26], the authors built on this by introducing a migratory shared protection scheme. This approach prepares backup nodes and allows virtual nodes to migrate to backup nodes that are closer to other virtual nodes, effectively reducing the length of the backup path and optimizing resource utilization.

In [27], the authors further advance survivability by extending the single path approach to multiple disjoint primary paths with uniform bandwidth requirements. A single backup path is configured to protect these disjoint primary paths, ensuring that

doi:10.6342/NTU202501591

when a single substrate link fails, the unaffected primary paths and the backup path can maintain service continuity. This design reduces the required backup bandwidth compared to earlier methods. In [28], the authors enhance the multi-path splitting approach proposed in [27] by integrating resource sharing techniques to further reduce backup resource requirements. It combines path splitting with two types of resource sharing: intra-sharing and inter-sharing. Intra-sharing occurs within a single virtual network request, where backup resources for different virtual links are shared to minimize redundancy. Inter-sharing allows backup resources to be shared across multiple VNRs by overlapping backup paths strategically while ensuring survivability. These enhancements enable [28] to achieve greater efficiency in backup resource utilization compared to [27].

2.2.2 Node Failure in Virtual Network Embedding

Node failures in virtual network embedding are divided into switching node failure and facility node failure. Switching node failure not only disrupt the failed node but also cause the failure of its adjacent substrate links, introducing the additional challenge of addressing link failures alongside node failures. In contrast, facility node failure only affects the failed node, requiring solutions that focus solely on node survivability.

2.2.2.1 Switching Node Failure

In [10], the authors address switching node failures by re-mapping affected virtual nodes to functioning substrate nodes and restoring connectivity through efficient path re-mapping algorithms. The study utilizes the Max-Flow Algorithm to identify feasible paths for multiple affected virtual links simultaneously, optimizing bandwidth

doi:10.6342/NTU202501591

utilization and avoiding the bottlenecks caused by bandwidth insufficiency that can arise in sequential re-mapping processes. Additionally, the Shortest Path Algorithm is employed to restore affected virtual links when only path failures occur due to a switching node failure.

In [11], the authors address the challenge of restoring VN under switching node failures. To ensure recovery, the study focuses on selectively re-mapping only the affected virtual nodes and links, while leaving the unaffected parts of the VN unchanged. By pre-allocating redundant resources, including computing capacity and bandwidth, this approach ensures efficient recovery and maintains service continuity.

In [12], the authors propose a dual-mapping approach for virtual networks to ensure survivability against switching node failures. This method maps the virtual network into two completely disjoint copies: a primary virtual network for normal operations and a backup virtual network for recovery. The substrate nodes and links used for the primary and backup networks are entirely separate, enabling the approach to effectively handle both mapped node and link failures. In the event of a switching node failure, all operations are migrated from the primary resources to the backup resources, ensuring immediate recovery and uninterrupted service continuity.

2.2.2.2 Facility Node Failure

Facility node failure assumes that only the failed substrate node becomes non-functional while its adjacent substrate links remain intact. This contrasts with switching node failure, in which both the node and its adjacent links fail, requiring both node and link recovery strategies. Due to this distinction, the studies discussed in

doi:10.6342/NTU202501591

Section 2.2.2.1 on switching node failure ([10], [11], and [12]) can also be applied to address the SVNE problem in facility node failure scenarios.

Studies such as [10], [11], and [12] propose solutions that directly address the SVNE problem by embedding virtual networks onto the SN while simultaneously ensuring survivability. However, the large scale of SN makes it computationally infeasible to efficiently solve such an NP-hard problem. This inherent complexity limits the practical applicability of these methods, as they struggle to balance efficiency and optimality in real-world scenarios. To address these challenges, some studies have proposed a decoupled approach to tackle survivability and mapping separately [13]-[17]. Specifically, these approaches first augment the VN by integrating survivability requirements into the VN. Then, the augmented VN is mapped onto the substrate network during the mapping (embedding) phase, focusing solely on solving the mapping problem. This decoupled strategy offers the following advantages.

1. **Problem Size Reduction:**

   As the size of VN is generally smaller than that of the substrate network, solving the survivability problem at the VN level significantly reduces problem complexity.

2. **Increased Flexibility in the Mapping (Embedding) Phase:**

   Since the survivability requirements are already addressed during the VN augmentation phase, the mapping (embedding) phase can focus on optimizing a single objective, such as resource utilization efficiency or VN acceptance rate. This allows for specialized algorithms to achieve lower costs and higher efficiency in resource usage.

doi:10.6342/NTU202501591

To address survivability, the work in [13] proposes two redundancy-based approaches for ensuring survivability: 1-redundant and N-redundant methods. The 1-redundant method assigns a single backup node to protect all virtual nodes, whereas the N-redundant method allocates N backup nodes, each dedicated to protecting a specific virtual node, where N represents the number of primary virtual nodes in the VN. In the mapping (embedding) phase, the primary virtual resources, including virtual nodes and virtual links, are first mapped using existing VNE mapping algorithms [2]. Subsequently, it solves a Mixed-Integer Linear Programming (MILP) problem to map the backup resources, including backup virtual nodes and backup virtual links, onto the SN. This step combines the embedding and survivability problems into a single optimization problem on the SN.

The study in [14] employs an N/2-redundant approach that augments the VN by adding backup nodes equivalent to half the number of primary virtual nodes, specifically assigned to protect the top N/2 primary virtual nodes with the highest degrees. The remaining primary virtual nodes, however, are not initially protected in this phase. Additionally, the VN is augmented with the necessary backup links to support the survivability design. For embedding, primary virtual resources are mapped using the PSO algorithm from [29], and the backup resources are mapped using the same algorithm to ensure the survivability of all primary virtual nodes during the mapping process. This approach safeguards the previously unprotected primary virtual nodes by assigning specific backup nodes during the embedding phase.

The study in [15] proposes a method to ensure survivability by assigning a dedicated backup node to protect each primary virtual node in the VN and introducing

doi:10.6342/NTU202501591

corresponding backup links. Unlike [13] and [14], where primary and backup resources are mapped in separate phases, this unified process utilizes a Tabu-search algorithm to solve the combined mapping problem.

Although previous studies [13]-[15] proposed methods for constructing augmented VNs by increasing backup resources to enhance survivability, they did not analyze how the amount of backup resources impacts the mapping (embedding) cost when the augmented VN is mapped onto the SN. As a result, although their approaches aimed to improve survivability at the VN level, they did not lead to cost-effective solutions for the SVNE problem when embedded in the SN. This gap highlights the need for further investigation into the relationship between backup resource increment and mapping (embedding) costs to develop more efficient SVNE strategies.

In contrast, studies in [16] and [17] assumed that minimizing the resource increment during the augmentation phase would lead to a lower mapping (embedding) cost, ultimately achieving a globally optimal solution. Their methodologies, therefore, focused on reducing the resource increment required for survivability, aiming to achieve cost efficiency in the mapping (embedding) phase.

In [16], the study proposes two protection mechanisms: Failure Independent Protection (FIP) and Failure Dependent Protection (FDP). These mechanisms define how virtual nodes are handled during a single facility node failure. FIP allows only the affected virtual node to migrate, with each migrating to a pre-designated backup node. In contrast, FDP permits both affected and unaffected virtual nodes to migrate,

doi:10.6342/NTU202501591

allowing any virtual node to relocate to either an existing backup node or another functioning virtual node's original location that has become vacant. This approach allows for more efficient utilization of backup resources, thereby minimizing the required backup resource increments.

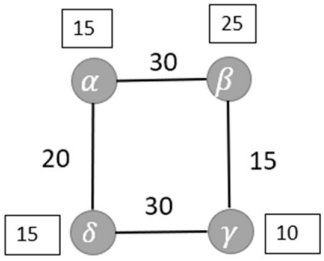


Fig.1 A virtual network (VN)

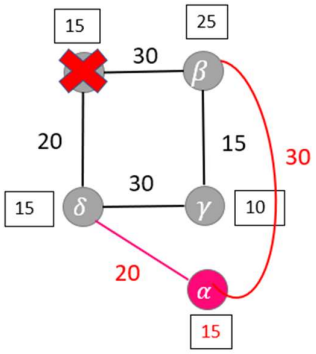


Fig.2 FIP: α affected by node failure

In the FIP method, only the affected virtual node is allowed to migrate in the event of a failure. For example, consider Fig. 1, which illustrates an VN. To augment this VN and enhance its survivability, the augmentation process is applied as shown in Fig. 2. In Fig. 2, if the virtual node α is affected by a failure, it is migrated to a designated backup node. To restore the transmission service of α, virtual backup links are provisioned between α and other relevant nodes. The backup resources allocated specifically for α in case of failure are highlighted in red, representing the additional computational and bandwidth requirements necessary for ensuring survivability.

doi:10.6342/NTU202501591

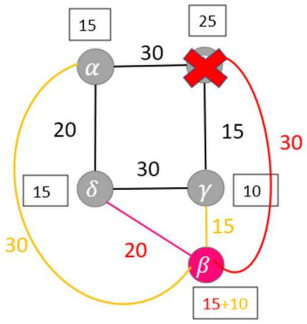


Fig.3 FIP: β affected by node failure

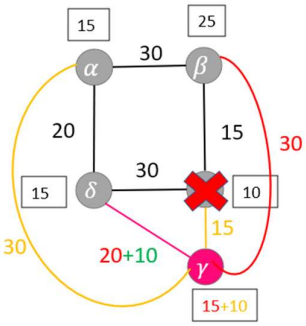


Fig.4 FIP: γ affected by node failure

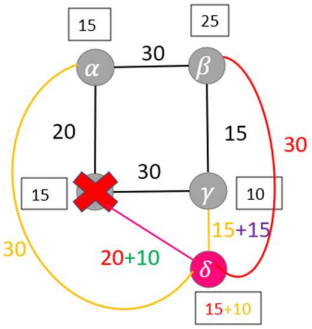


Fig.5 FIP: δ affected by node failure

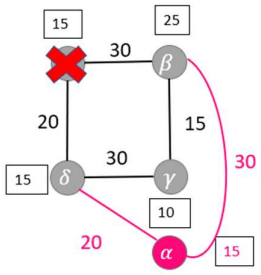
Figures 3, 4, and 5 illustrate the sequential process of handling failures for virtual nodes β, γ, and δ, respectively. When each virtual node is affected, it is migrated to a designated backup node, and virtual backup links are established to restore its transmission service.

To optimize resource usage, backup resources can be shared across different failure scenarios, effectively reducing the overall increment in backup resource. Specifically, the additional backup resources allocated to protect β, γ, and δ are highlighted in orange, green, and purple text, respectively. These color-coded annotations represent the respective increases in computational and bandwidth resources necessary to maintain survivability.

doi:10.6342/NTU202501591

Finally, the total increment for augmenting this VN using the FIP method is calculated. The node resource increment is 25, the bandwidth increment is 120, and the total resource increment of the augmented VN is 145.

In the FDP method, both affected and unaffected virtual nodes are allowed to migrate in the event of a failure. To efficiently identify migration strategies that minimize resource increments while ensuring survivability, an increment matrix is constructed. In this matrix, the columns represent virtual nodes, and the rows represent functioning vacant nodes to which these virtual nodes can migrate. These functioning vacant nodes can either be backup node or the original locations of other virtual nodes.

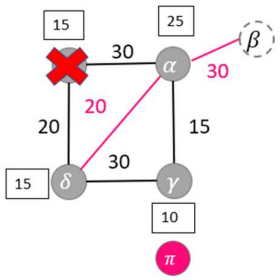


origin/migration	*backup*	*β*	*γ*	*δ*
α	65			
β				
γ				
δ				

Fig.6 FDP: α migrates to backup, increment recorded in increment matrix

For instance, consider the case where virtual node α is affected. Fig. 6 illustrates the augmentation process for migrating α to a backup node. The total resource increment required for this migration amounts to 65 units, encompassing both computational and bandwidth resources.

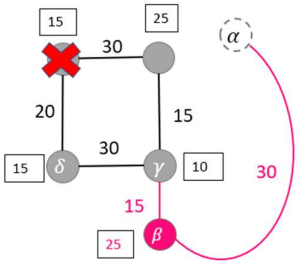
doi:10.6342/NTU202501591



origin/migration	backup	β	γ	δ
α	65	50	☐	☐
β	☐	☐	☐	☐
γ	☐	☐	☐	☐
δ	☐	☐	☐	☐

Fig.7 FDP: α migrates to β's vacant node, increment recorded in increment matrix

Virtual node α can also be migrated to the original locations of other virtual nodes apart from its designated backup node. For instance, as shown in Fig. 7, α is migrated to the vacant node originally allocated to virtual node β. In this scenario, β must subsequently be relocated to another vacant node to accommodate α's migration. However, to avoid increasing the complexity of the increment matrix, the actual relocation position of β is not considered. Instead, only the migration cost associated with α utilizing the vacant node is accounted for in the calculation. In this case, the total resource increment required amounts to 50 units.



origin/migration	backup	β	γ	δ
α	65	50	20	50
β	70	☐	☐	☐
γ	☐	☐	☐	☐
δ	☐	☐	☐	☐

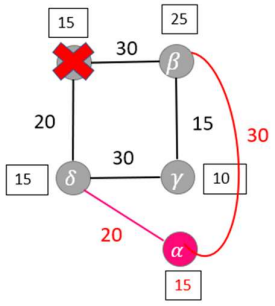
Fig.8 FDP: migrates to other vacant nodes, increment recorded in increment matrix

Using the same approach, the incremental resource requirements for migrating α to two other functioning vacant nodes are calculated, requiring 20 and 50 additional resources, respectively, as shown in Fig. 8. Furthermore, other unaffected virtual

doi:10.6342/NTU202501591

nodes can also migrate. For instance, as illustrated in Fig. 8, when β migrates to its designated backup node, the total resource increment required for this migration is 70. As previously mentioned, to reduce the complexity of the matrix computation, only the migration cost of β is considered, without accounting for the actual position of α.

Using the same method, the incremental resource requirements for all potential migration locations of all virtual nodes are calculated under the scenario where the virtual node α is affected. Following the principle that each virtual node must be assigned to a distinct functioning vacant node, the optimal migration strategy with the minimum resource increment is identified to ensure service continuity.

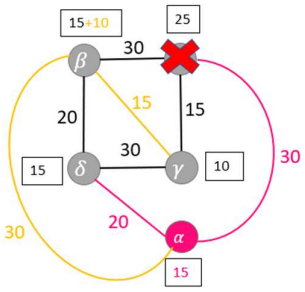


origin/migration	*backup*	*β*	*γ*	*δ*
α	**65**	50	20	50
β	70	**30**	60	40
γ	55	45	**0**	45
δ	65	35	55	**20**

Fig.9 FDP: α affected with minimum bipartite matching

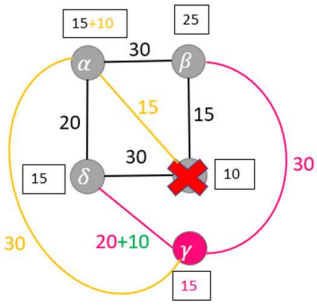
To achieve this, the Hungarian algorithm is applied to the increment matrix to determine the minimum-increment bipartite matching. As illustrated in Fig. 9, the optimal bipartite matching identified involves migrating only α to its backup node. The results of the bipartite matching are highlighted with a green background to indicate the selected allocations.

doi:10.6342/NTU202501591



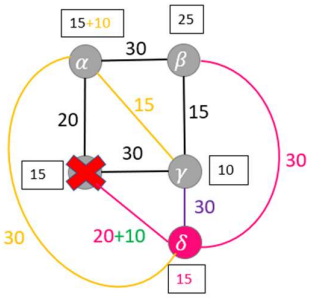
origin/migration	α	*backup*	γ	δ
α	30	**30**	35	50
β	**55**	55	60	20
γ	25	25	**15**	45
δ	50	50	55	**0**

Fig.10 FDP: β affected with minimum bipartite matching



origin/migration	α	β	*backup*	δ
α	**0**	50	0	50
β	45	**15**	25	35
γ	10	45	**10**	45
δ	50	30	30	**30**

Fig.11 FDP: γ affected with minimum bipartite matching



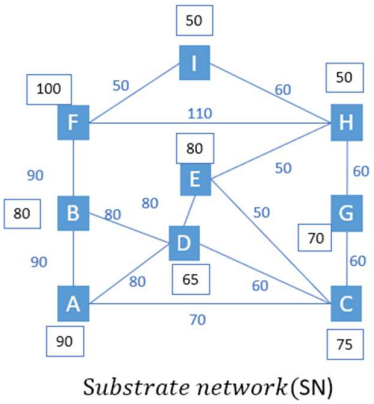
origin/migration	α	β	γ	*backup*
α	**20**	50	40	20
β	30	**0**	45	25
γ	30	45	**30**	30
δ	35	15	40	**30**

Fig.12 FDP: δ affected with minimum bipartite matching

Next, for each potential single node failure scenario, the same method is applied to construct increment matrices and calculate the migration costs for all virtual nodes. Using the Hungarian algorithm, the minimum-increment bipartite matching is determined for each increment matrix. As shown in Figs. 10, 11, and 12, the optimal

doi:10.6342/NTU202501591

bipartite matchings are identified for the cases where the virtual nodes of β, γ, and δ is affected, respectively. The additional backup resource increments required under these failure scenarios are highlighted in orange, green, and purple text, respectively, to distinguish the contributions for each failure scenario. Finally, the total increment for augmenting this VN using the FDP method is calculated. The node resource increment is 25, the bandwidth increment is 135, and the total resource increment of the augmented VN is 160.



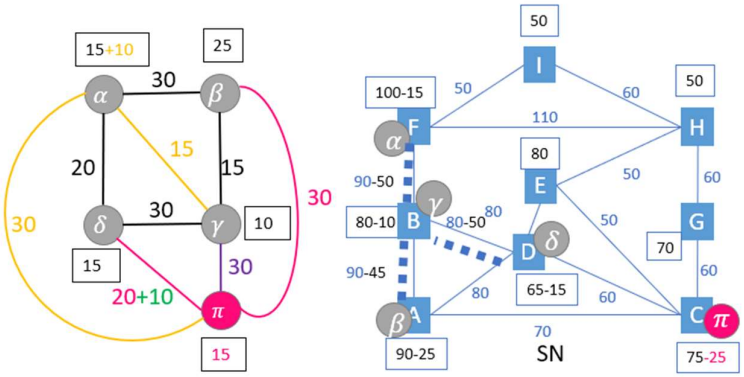
Substrate network(SN)

Substrate node	metric
A	90*(90+80+70)=21600
B	80*(90+80+90)=20800
C	75*(70+60+50+60)=18000
D	65*(80+80+80+60)=19500
E	80*(80+50+50)=14400
F	100*(50+90+110)=25000
G	70*(60+60)=8400
H	50*(60+110+50+60)=14000
I	50*(50+60)=5500

Fig.13 SN and substrate node metric values

After completing the augmentation of the VN, the augmented VN is mapped onto the SN. The mapping process begins with node mapping, where virtual nodes are sequentially mapped to substrate nodes in an arbitrary order that satisfy the computational resource requirements and maximize the metric value. The metric is defined as the product of the available computational resources of a substrate node and the total available bandwidth of its adjacent substrate links, as illustrated in Fig. 13.

doi:10.6342/NTU202501591



Virtual link	Major Path	Bandwidth demand
$\alpha - \beta$	ABF	30
$\beta - \gamma$	AB	15
$\gamma - \delta$	DB	30
$\delta - \alpha$	DBF	20

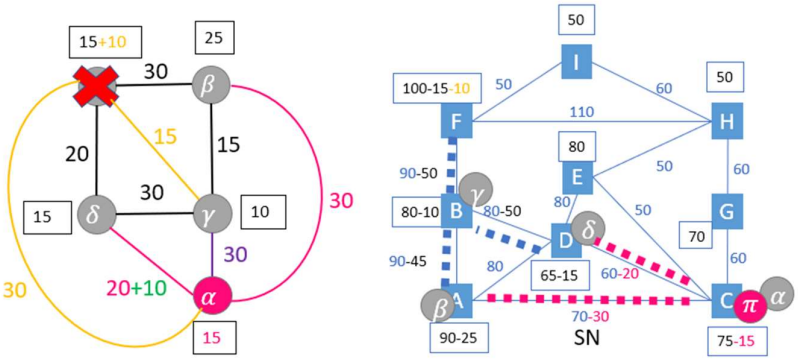
Fig.14 Mapping of primary nodes and links

Following this process, and taking the augmented VN generated under the FDP method as an example, virtual nodes α, β, γ, δ, and the backup node are mapped to substrate nodes F, A, B, D, and C, respectively, as shown in Fig. 14. After node mapping is completed, the primary virtual links are then mapped using the shortest path algorithm, also depicted in Fig. 14.

Next, the mapping process continues with the allocation of backup paths. It is important to note that during the augmentation of the VN, the consideration of backup bandwidth sharing does not account for the potential sharing of bandwidth among different backup links. To minimize the usage of backup bandwidth on the SN, the backup paths are not mapped according to the predefined augmented VN. Instead, the process considers one virtual node is affected at a time and identifies backup paths to

doi:10.6342/NTU202501591

restore the normal operation of the VN. This approach maximizes the potential for bandwidth sharing across the SN, reducing the overall consumption of backup resources.

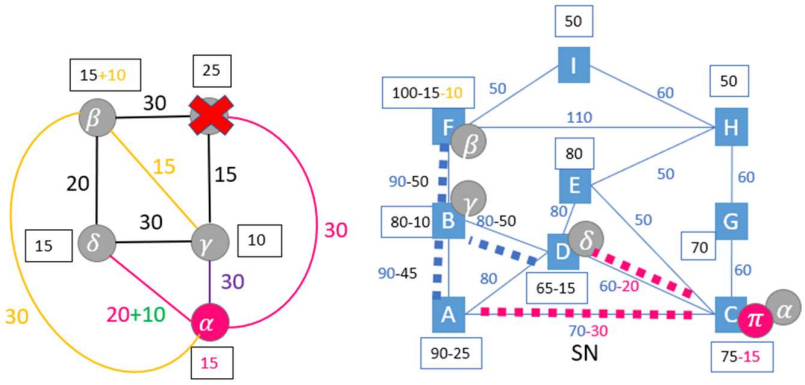


Virtual link	Major Path	Backup Path	Bandwidth demand
$\alpha - \beta$	ABF	CA	30
$\beta - \gamma$	AB	-	15
$\gamma - \delta$	DB	-	30
$\delta - \alpha$	DBF	CD	20

Fig.15 FDP: Backup resource mapping for α affected

To address the scenario where α is affected, α is migrated to the backup node located on substrate node C. Subsequently, a weighted shortest path algorithm is employed to identify and map the backup paths required to restore connectivity, as illustrated in Fig. 15.

doi:10.6342/NTU202501591

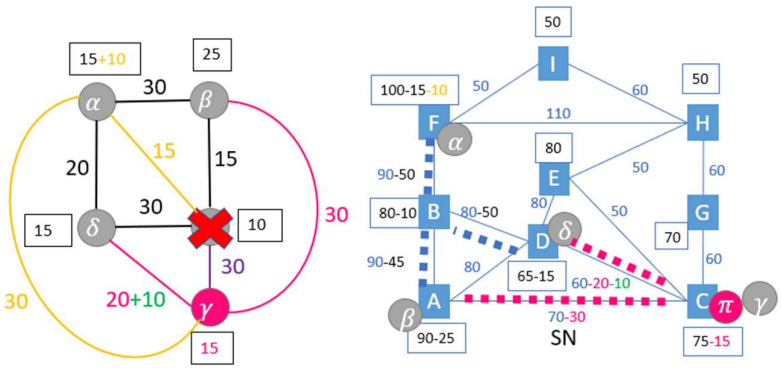


Virtual link	Major Path	Backup Path	Bandwidth demand
$\alpha - \beta$	ABF	CABF	30
$\beta - \gamma$	AB	BF	15
$\gamma - \delta$	DB	-	30
$\delta - \alpha$	DBF	CD	20

Fig.16 FDP: Backup resource mapping for β affected

When virtual node β is affected, following the virtual node migration strategy defined in the augmented VN, α is migrated to the backup node located on substrate node C, and β is migrated to substrate node F, which previously hosted α. To minimize bandwidth costs, shared backup paths and the reuse of primary paths released during the virtual node migrations are utilized. Additionally, the weighted shortest path algorithm is employed to identify the least-cost paths for backup mapping, as illustrated in Fig. 16.

doi:10.6342/NTU202501591

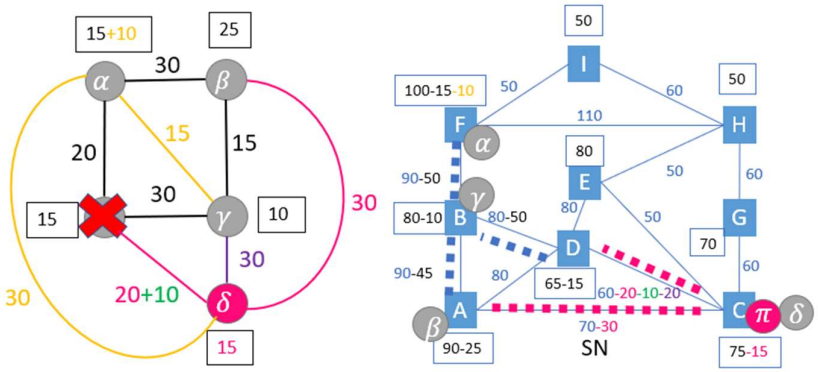


Virtual link	Major Path	Backup Path	Bandwidth demand
$\alpha - \beta$	ABF	-	30
$\beta - \gamma$	AB	CA	15
$\gamma - \delta$	DB	CD	30
$\delta - \alpha$	DBF	-	20

Fig.17 FDP: Backup resource mapping for γ affected

When virtual node γ is affected, following the virtual node migration strategy defined in the augmented VN, γ is migrated to the backup node located on substrate node C. As before, the weighted shortest path algorithm is utilized to identify the least-cost paths, ensuring minimal bandwidth consumption, as illustrated in Fig. 17.

doi:10.6342/NTU202501591



Virtual link	Major Path	Backup Path	Bandwidth demand
$\alpha - \beta$	ABF	-	30
$\beta - \gamma$	AB	-	15
$\gamma - \delta$	DB	CDB	30
$\delta - \alpha$	DBF	CDBF	20

Fig.18 FDP: Backup resource mapping for δ affected

When virtual node δ is affected, following the virtual node migration strategy defined in the augmented VN, δ is migrated to the backup node located on substrate node C. Similarly, the weighted shortest path algorithm is employed to identify the least-cost paths, ensuring that the backup path mapping minimizes bandwidth consumption, as illustrated in Fig. 18.

Since all primary virtual nodes are protected through the backup node and backup paths, this solution ensures survivability in the event of a single facility node failure. The total mapping cost is then calculated as follows: the computational cost is 90, the bandwidth cost is 225, resulting in a total mapping cost of 90+225=315.

doi:10.6342/NTU202501591

In [17], the study introduces an innovative approach to enhance survivability by strategically placing backup node on the virtual links between two primary virtual nodes. This placement allows virtual node migrating to the backup node to efficiently reuse released bandwidth resources, thereby optimizing resource utilization during virtual node migration. Additionally, compared to [16], [17] allows for multiple backup nodes within the augmented virtual network but restricts migration to affected virtual node only. This design improves the efficiency of backup bandwidth and the utilization of released primary bandwidth, ultimately reducing the required backup bandwidth increment and further minimizing resource overhead.

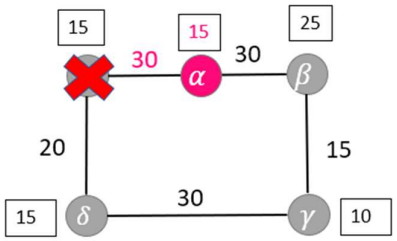


Fig.19 α affected, migrates to backup node

The authors propose a heuristic method called Prognostic Redesign Heuristic (ProRed) for augmenting the VN. Taking the VN in Fig. 1 as an example, the heuristic begins by selecting the primary virtual node with the highest degree for protection. In this case, since all four primary virtual nodes have the same degree, any node can be selected arbitrarily. For instance, if α is chosen, a backup node is placed on the primary link between α and β. This placement is strategic, as the primary link between α and β has the highest bandwidth requirement 30 among α's adjacent links. When the virtual node α is affected, α is migrated to the backup node. This approach efficiently reuses the released primary bandwidth to re-establish connections with δ and β, ensuring service continuity while minimizing additional bandwidth requirements, as shown in Fig. 19.

doi:10.6342/NTU202501591

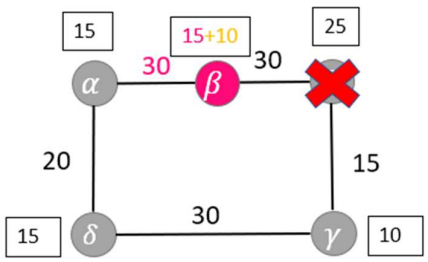


Fig.20 β affected, migrates to backup node

Next, the heuristic proceeds to protect the primary virtual node β, which is adjacent to the backup node. Similarly, when the virtual node β is affected, β is migrated to the backup node. This migration efficiently reuses the released primary bandwidth to re-establish connections with α and γ, as illustrated in Fig. 20.

To minimize the increase in backup bandwidth, if the virtual node γ or δ is affected, using the backup node placed between α and β to protect γ or δ would require additional backup bandwidth on the link between α and β. This is because γ or δ can only restore their services through the existing links, which are already utilizing the bandwidth between α and β. Therefore, additional backup bandwidth would be necessary.

To address this limitation, a second backup node is introduced to protect γ and δ. This second backup node is placed on the primary link between γ and δ, allowing both γ and δ to share this backup node effectively. Since the backup node between α and β exclusively protects α and β, a set is created to group α, β, and their shared backup node, representing that all primary virtual nodes within this set are protected by the corresponding backup node.

doi:10.6342/NTU202501591

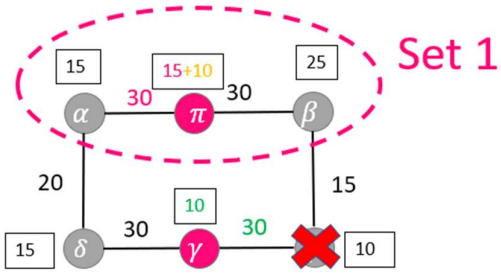


Fig.21 γ affected, migrates to backup node

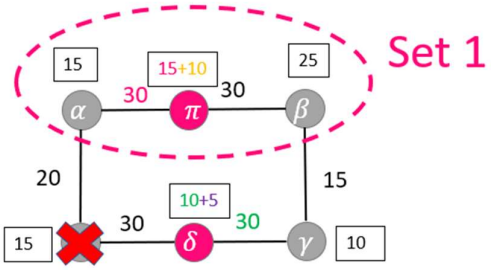


Fig.22 δ affected, migrates to backup node

As illustrated in Fig. 21 and Fig. 22, the scenarios depict the migration of γ or δ to the second backup node when γ or δ is affected. By utilizing the second backup node, γ and δ can effectively reuse the released bandwidth, significantly reducing the required increment in bandwidth resources.

Furthermore, since the second backup node provides protection for both γ and δ, they are grouped together into a set, representing that the primary virtual nodes within this set are collectively protected by the second backup node.

doi:10.6342/NTU202501591

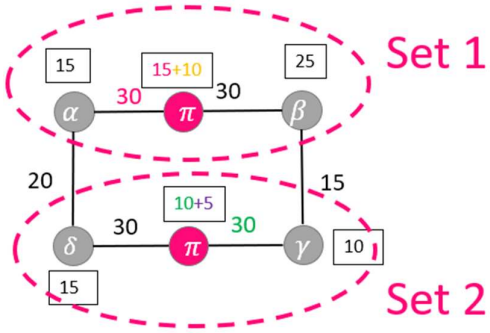


Fig.23 ProRed: Augmented VN

With all the primary virtual nodes protected by backup nodes, the augmented VN design is completed, as shown in Fig. 23. Finally, the resource increments are calculated, with the backup computational resource increment being 40 and the backup bandwidth resource increment being 60, resulting in a total resource increment of 100.

doi:10.6342/NTU202501591

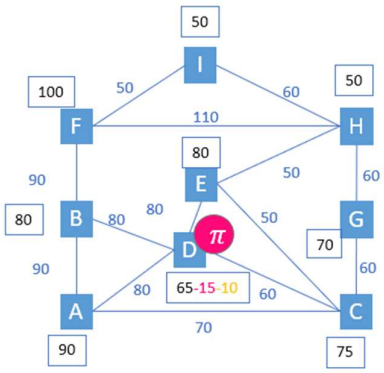


Fig.24 First node mapping of augmented VN

Substrate node	metric
A	3*(90+80+70)=720
B	3*(90+80+90)=780
C	4*(70+60+50+60)=960
D	4*(80+80+80+60)=1200
E	3*(80+50+50)=540
F	3*(50+90+110)=750
G	2*(60+60)=240
H	4*(60+110+50+60)=1120
I	2*(50+60)=220

Fig.25 Substrate node metric values

After completing the augmentation of the VN using the ProRed algorithm, the next step involves mapping the augmented VN onto the SN. The mapping process begins by selecting the backup node within the augmented VN that is part of the set with the largest adjacent bandwidth requirement, such as the backup node from set 1. This backup node is then mapped onto the substrate node with the highest metric value that also satisfies the computational resource requirements, for example, substrate node D, as illustrated in Fig. 24. The metric value is calculated similarly to the approach shown in Fig. 13, where the degree of the substrate node is multiplied by the total available bandwidth of its adjacent substrate links, as shown in Fig. 25.

doi:10.6342/NTU202501591

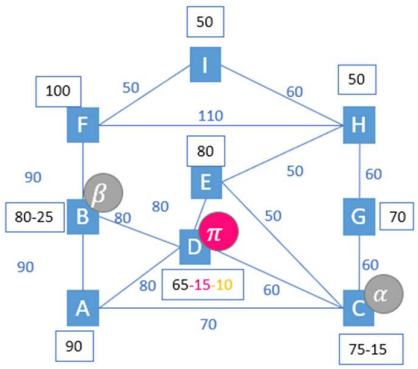


Fig.26 Mapping of nodes adjacent to backup node

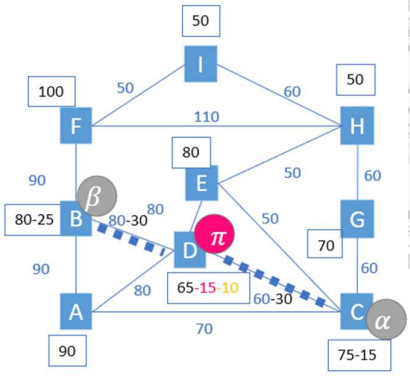


Fig.27 Link mapping

Next, all virtual nodes in set 1 are mapped sequentially, with the order determined by the bandwidth requirements between each node and the already-mapped backup node. These requirements are sorted in descending order. Since α and β both have the same bandwidth requirement with the backup node, either can be chosen first. For example, α is mapped first, followed by β. Virtual node α is mapped to the substrate node closest to the substrate node where the backup node is mapped (node D) and with the highest metric value, as shown in Fig. 26, such as substrate node C. Similarly, β is mapped to the substrate node closest to node D and with the highest metric value, such as substrate node B, as illustrated in Fig. 26.

Since there is a primary virtual link between α and β in the original VN, as shown in Fig. 1, the primary virtual link between these two virtual nodes is then mapped to the substrate network, as shown in Fig. 27.

doi:10.6342/NTU202501591

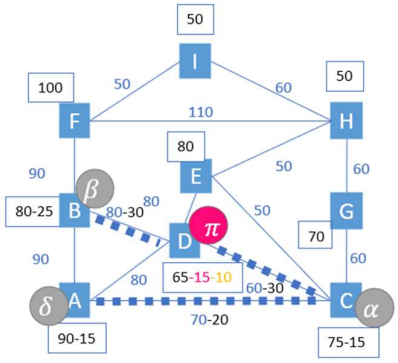


Fig.28 Mapping of node adjacent to α

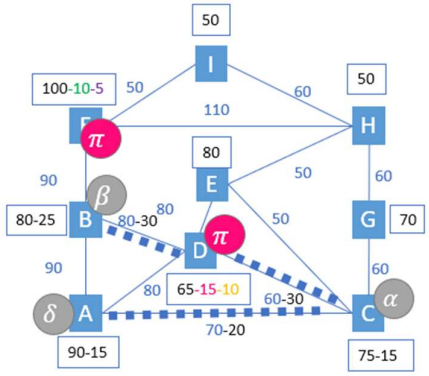


Fig.29 Mapping of node adjacent to δ

After completing the mapping of all virtual nodes in set 1, the next step is to map the virtual node with the highest bandwidth requirement for its primary virtual link connected to set 1. For instance, δ is selected. Similarly, δ is mapped to the substrate node closest to the substrate node where α is mapped (node C) and with the highest metric value, such as substrate node A. Since there is a primary virtual link between δ and α in the original VN, as shown in Fig. 1, this primary virtual link is also mapped to the substrate network. The mapping process and resulting connections are illustrated in Fig. 28.

Next, the remaining virtual nodes in set 2 are mapped, starting with the backup node followed by γ, as the bandwidth requirement between the backup node and already mapped nodes is higher. The backup node is mapped to the substrate node closest to the substrate node where δ is mapped (node A) and with the highest metric value, such as substrate node F, as shown in Fig. 29.

doi:10.6342/NTU202501591

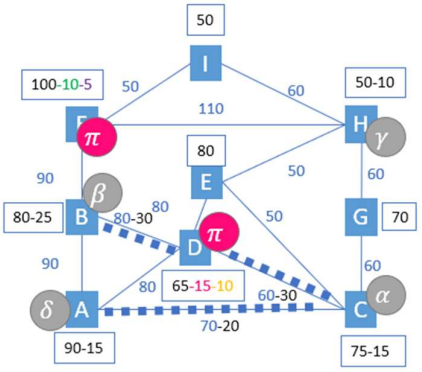


Fig.30 Mapping of nodes adjacent to backup node

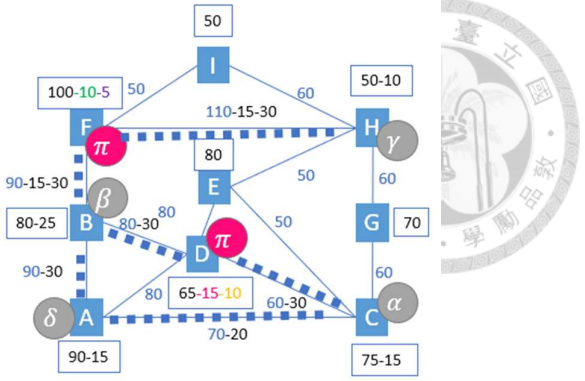
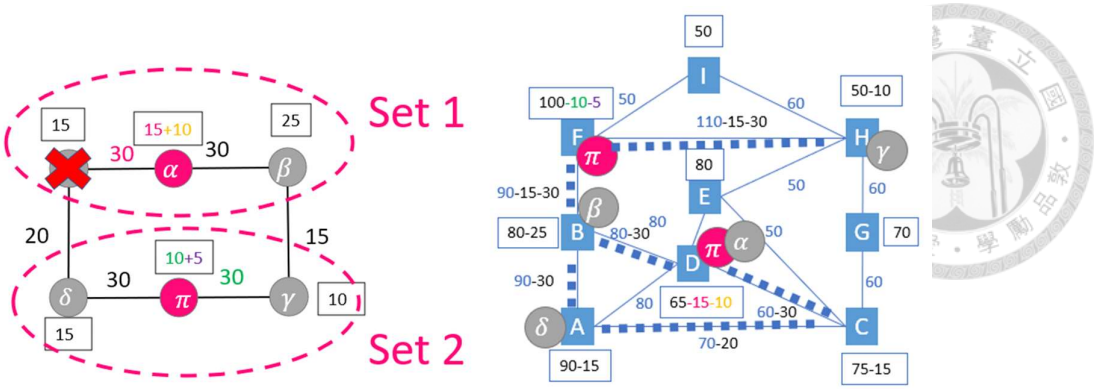


Fig.31 Link mapping

Finally, γ is mapped to the substrate node closest to both the substrate node where the backup node is mapped (node F) and the substrate node where β is mapped (node B), while also having the highest metric value, such as substrate node H, as shown in Fig. 30. Subsequently, the primary paths between γ and β, as well as between γ and δ, are mapped, as illustrated in Fig. 31.

Once all primary and backup virtual nodes, as well as the primary virtual links, have been mapped, the next step is to handle the mapping of the backup virtual paths. Since the mapping of the primary virtual paths aimed to minimize bandwidth mapping costs, it did not account for whether the primary paths passed through backup nodes. As a result, the backup virtual paths cannot fully adhere to the structure of the augmented VN. Instead, the process considers one virtual node is affected at a time and identifies backup paths to restore the normal operation of the VN. This approach maximizes the reuse of released primary path bandwidth while ensuring survivability.

doi:10.6342/NTU202501591

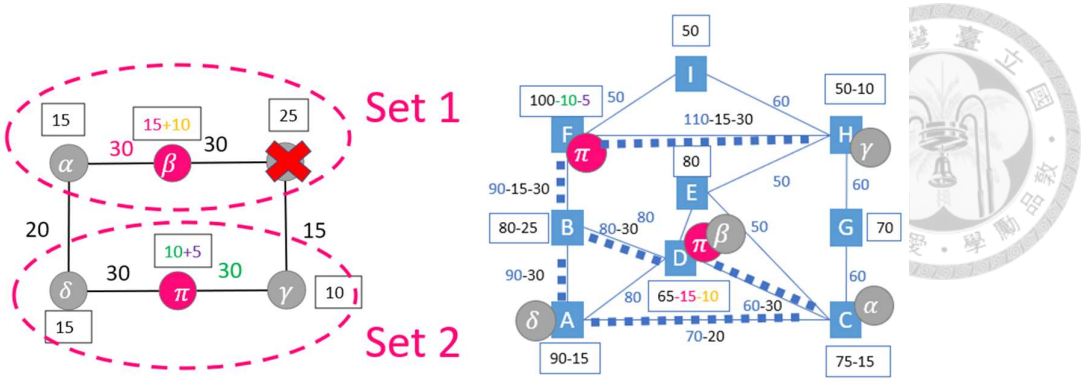


Virtual link	Major Path	Backup Path	Bandwidth demand
$\alpha - \beta$	CDB	DB	30
$\beta - \gamma$	BFH	-	15
$\gamma - \delta$	ABFH	-	30
$\delta - \alpha$	CA	DCA	20

Fig.32 Backup links mapping for α affected

The process sequentially considers one virtual node is affected at a time, in an arbitrary order. For example, it handles the failure scenarios where virtual nodes α, β, γ, and δ are affected one at a time, sequentially. First, when the virtual node α is affected, the virtual node α is migrated to the backup node located at substrate node D. The backup paths are then established, leveraging the released primary path bandwidth to restore connectivity efficiently, as illustrated in Fig. 32.

doi:10.6342/NTU202501591

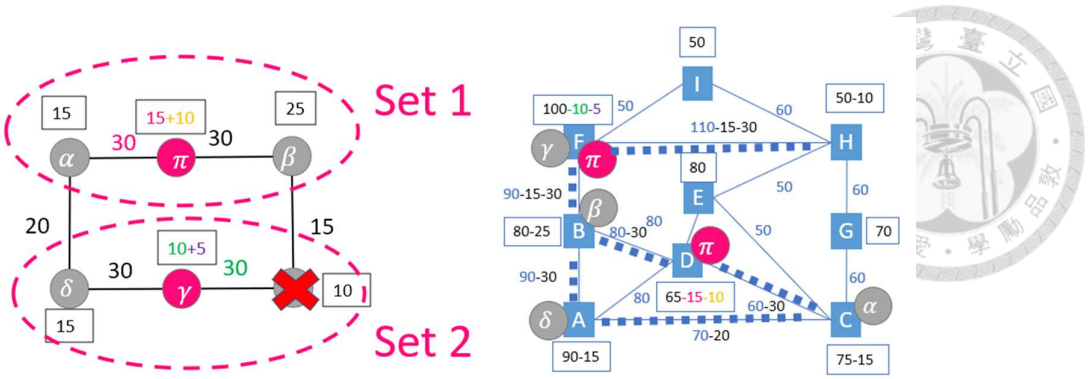


Virtual link	Major Path	Backup Path	Bandwidth demand
$\alpha - \beta$	CDB	CD	30
$\beta - \gamma$	BFH	DBFH	15
$\gamma - \delta$	ABFH	-	30
$\delta - \alpha$	CA	-	20

Fig.33 Backup links mapping for β affected

Next, when the virtual node β is affected, the virtual node β is migrated to the backup node located at substrate node D. Backup paths are then established, utilizing the released primary path bandwidth to restore connectivity effectively, as shown in Fig. 33.

doi:10.6342/NTU202501591

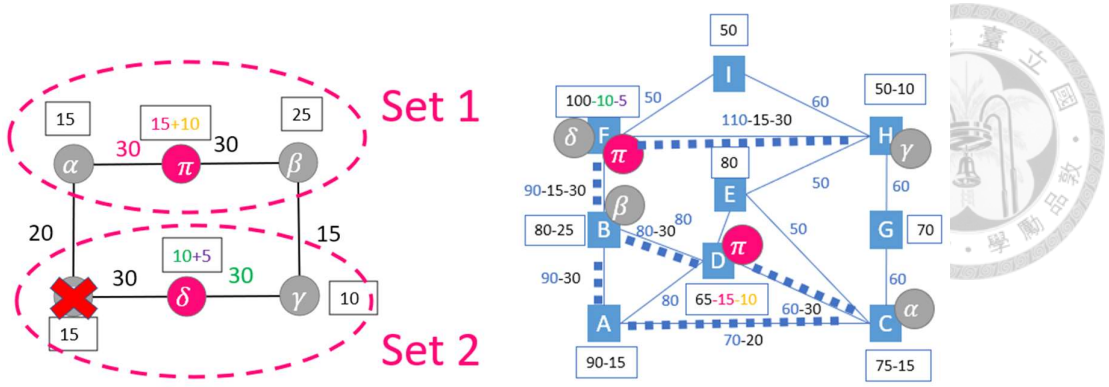


Virtual link	Major Path	Backup Path	Bandwidth demand
$\alpha - \beta$	CDB	-	30
$\beta - \gamma$	BFH	BF	15
$\gamma - \delta$	ABFH	ABF	30
$\delta - \alpha$	CA	-	20

Fig.34 Backup links mapping for γ affected

Next, when the virtual node γ is affected, the virtual node γ is migrated to the backup node located at substrate node F. Backup paths are established by utilizing the released primary path bandwidth to restore connectivity effectively, as shown in Fig. 34.

doi:10.6342/NTU202501591



Virtual link	Major Path	Backup Path	Bandwidth demand
$\alpha - \beta$	CDB	-	30
$\beta - \gamma$	BFH	-	15
$\gamma - \delta$	ABFH	FH	30
$\delta - \alpha$	CA	CABF	20

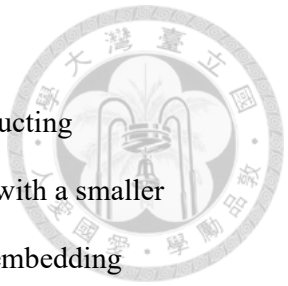
Fig.35 Backup links mapping for δ affected

Finally, when the virtual node δ is affected, the virtual node δ is migrated to the backup node located at substrate node F. Backup paths are established by leveraging the released primary path bandwidth to efficiently restore connectivity, as illustrated in Fig. 35.

With the completion of the augmented VN mapping, this solution ensures survivability in the event of a single facility node failure. The total mapping cost is then calculated as follows: the computational cost is 105, the bandwidth cost is 200, resulting in a total mapping cost of 105 + 200 = 305. Notably, in this example, no additional backup bandwidth needs to be reserved, demonstrating the perfect sharing of backup resources.

doi:10.6342/NTU202501591

2.3 Identified Problems of [16] and [17]



Although the two studies [16],[17] propose different methods for constructing augmented VNs, they share a common assumption: the augmented VN with a smaller resource increment during the augmentation phase will lead to a lower embedding cost during the mapping (embedding) phase. As a result, the augmented VN with the smallest increment is expected to achieve the globally optimal embedding cost. However, we have found that such a linear relationship between the increment in the augmentation phase and the mapping cost in the embedding phase does not necessarily exist. This issue is analyzed in Section 2.3.1, where we demonstrate that both [16] and [17] fail to ensure this linearity, as their heuristic approaches do not consistently correlate smaller increments with lower embedding costs.

Furthermore, [17] imposes stricter constraints on the VN augmentation process compared to [16], as discussed in Section 2.3.1. These constraints make the augmentation process in [17] less flexible, leading to a smaller solution space for possible augmented VNs. In Section 2.3.2, we further examine how this narrower solution space affects how low the minimum mapping cost can be, in comparison to [16].

2.3.1 Lack of Linearity in [16] and [17]

In [16], the study introduces two augmentation methods, Failure Independent Protection (FIP) and Failure Dependent Protection (FDP). FDP offers greater flexibility in augmentation strategies compared to FIP.

doi:10.6342/NTU202501591

FIP and FDP define two solution spaces of possible augmented VNs, each reflecting the constraints and flexibility in their respective virtual node migration rules. FIP is highly restrictive, assigning each affected virtual node to a pre-designated backup node, which leaves only one possible migration configuration per failure scenario, resulting in a solution space that contains only a single augmented VN. Conversely, FDP allows both affected and unaffected virtual nodes to migrate, leading to a significantly larger solution space of possible augmented VNs. For a VN with N = 4 virtual nodes, as shown in Fig. 1, each failure scenario permits 4! = 24 unique migration arrangements of the virtual nodes. Since the augmented VN's total backup resource increment is derived from the cumulative resource increments required across all failure scenarios, these migration options are independently considered for each of the N = 4 failure scenarios. As a result, the solution space under FDP contains $(N!)^N = (4!)^4 = 331776$ augmented VNs.

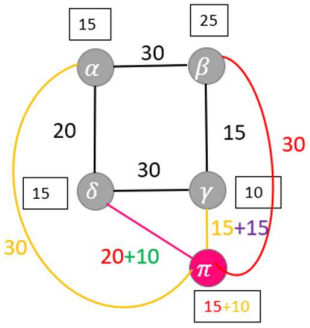


Fig.36 FIP: Augmented VN

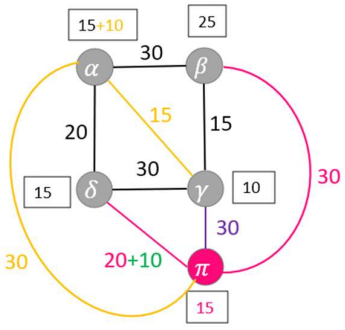


Fig.37 FDP: Augmented VN

Fig. 36 depicts the augmented VN produced using the FIP method, which results in a total resource increment of 145. In contrast, Fig. 37 illustrates the augmented VN generated through the FDP method, yielding a total resource increment of 160. Although the FDP method employs a more flexible augmentation strategy compared

doi:10.6342/NTU202501591

to FIP and is expected to yield an augmented VN with a smaller resource increment, it fails to achieve this. This discrepancy arises because the augmentation matrix used in FDP evaluates backup resources independently for each failure scenario, without considering the global optimality of the overall backup resource configuration.

On the other hand, the example presented in Section 2.2.2.2 highlights an interesting phenomenon: the augmented VN generated using the FIP method results in a smaller resource increment compared to that of the FDP method, yet its mapping cost is higher. This outcome contradicts the hypothesis proposed in [16], which assumes that augmented VNs with smaller resource increments will correspond to lower mapping costs.

This unexpected outcome raises questions about the relationship between resource increments in augmented VNs and their corresponding mapping costs. To further investigate, the linearity of the mapping algorithm proposed in [16] is examined.

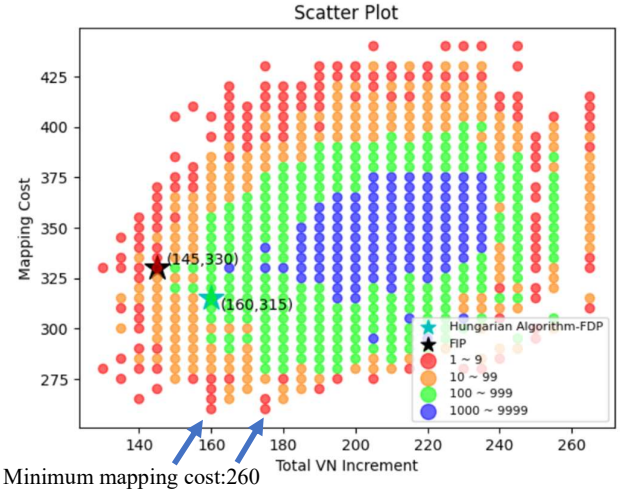


Fig.38 VN increment vs. mapping (embedding) cost for N=4 under FDP

doi:10.6342/NTU202501591

To examine the linearity of the mapping algorithm proposed in [16], all 331776 augmented VNs derived under the FDP method were mapped onto the same, independently designed SN. As shown in Fig. 38, each dot in the scatter plot represents a unique combination of VN increment and mapping cost, with its color indicating the number of distinct augmented VNs sharing that exact combination. For example, when the VN increment is 180 and the mapping cost is 300, the corresponding dot is green, meaning that at least 100 and at most 999 different augmented VNs share this exact combination. The specific number range for each color can be identified from the legend in the figure. The scatter plot illustrates the relationship between the augmented VN increment and the mapping cost.

The results indicate a lack of linearity, with the correlation coefficient measuring only 0.2834, reflecting insufficient linearity between the total VN increment and the mapping cost. This correlation coefficient was computed using all 331,776 data points, considering the relationship between the total augmentation increment and the corresponding mapping cost for each augmented VN.

A closer examination of Fig. 38 reveals that the augmented VN obtained through the FDP heuristic, which utilizes the augmentation matrix and the Hungarian algorithm, fails to identify the augmented VN with the smallest increment within its solution space. This is illustrated by the blue-green star, representing the FDP heuristic's result, which does not align with the lowest possible increment. Similarly, the black star marks the augmented VN generated by the FIP method. Interestingly, while the FDP method results in an augmented VN with a larger increment compared to FIP, its mapping cost is smaller. This discrepancy further underscores the inability of the

doi:10.6342/NTU202501591

mapping algorithm to align resource increments with mapping cost minimization effectively.

Similar to the methodology employed for [16], the augmented VNs generated under [17] were exhaustively enumerated. The augmentation rules in [17] collectively define a specific augmentation strategy, which involves placing backup nodes on virtual links within the VN and restricting migration during single node failure to only the affected virtual node, which must relocate to an available backup node. Furthermore, multiple backup nodes are permitted within the augmented VN, providing increased flexibility.

Taking the VN depicted in Fig. 1 as an example, the augmentation strategy defined by [17] allows the affected virtual node in each failure scenario to migrate to any backup node placed on the VN's virtual links. Under this strategy, all possible augmented VNs can be enumerated. For a VN with N = 4 virtual nodes and K = 4 virtual links, there are $K^N = 4^4 = 256$ possible augmented VNs in the solution space, as the total backup resource increment is determined by summing the backup resources required across all failure scenarios.

To further investigate, each of these 256 augmented VNs was mapped onto the same independently designed SN using the mapping algorithm proposed in [17]. The mapping results were visualized in a scatter plot (Fig. 39), which illustrates the relationship between the augmented VN's total backup resource increment and its corresponding mapping cost. Additionally, the augmented VN identified by the heuristic method, Prognostic Redesign Heuristic (ProRed), was highlighted with a

doi:10.6342/NTU202501591

blue-green star in the plot. Upon analyzing Fig. 39, a moderate linear trend between the augmented VN increment and mapping cost is observed. The correlation coefficient for this relationship is 0.7848, computed using all 256 data points, which considers the relationship between the total augmentation increment and the corresponding mapping cost for each augmented VN. While this indicates a certain degree of linearity, the correlation is not strong.

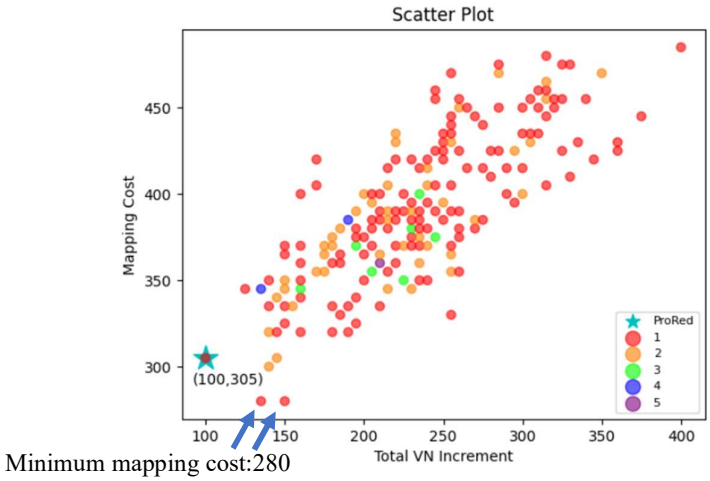


Fig.39 VN increment vs. mapping (embedding) cost for N=4 under ProRed

Moreover, while ProRed successfully identifies the augmented VN with the smallest total backup resource increment within the solution space of [17], it fails to achieve the lowest mapping cost, which is 280. This discrepancy further highlights the limitations of the mapping algorithm's linearity, as it does not effectively correlate smallest resource increments with minimized mapping costs.

The mapping algorithms proposed in [16] and [17] both exhibit insufficient linearity, a limitation evident in their resource allocation and utilization strategies. In the case of

doi:10.6342/NTU202501591

[16], the linearity issue arises from the consideration of backup path sharing during the mapping process. This introduces variability in backup resource mapping costs, which depend heavily on the degree of backup resource sharing in the SN. Specifically, when the shared backup resources are substantial, the mapping cost decreases; conversely, when the shared resources are minimal, the mapping cost increases. This variability disrupts the linear relationship between the total augmented VN increment and the resulting mapping cost, thereby compromising the algorithm's linearity.

In the case of [17], the inconsistency stems from the mapping process itself, which allows for backup resources to perfectly utilize the primary resources, as demonstrated in Section 2.2.2.2's example of Facility Node Failure. The root cause lies in the variability of path lengths for primary virtual links during mapping. Unlike [16], where backup resource sharing is the main issue, [17] suffers from the lack of control over the hop count of mapped virtual paths. When the path lengths extend beyond a single hop, the resulting mapping costs deviate significantly, severing the linear relationship between augmented VN increments and mapping costs. This highlights the critical challenge of ensuring consistent and predictable mapping paths in [17].

2.3.2 The Narrow Solution Space of [17]

From the analysis in Section 2.3.1, we observe that different augmentation strategies result in different lower bounds on the mapping cost. Specifically, under the augmentation strategy in [16], which corresponds to the FDP method, the lowest mapping cost is 260, as shown in Fig. 38. In contrast, under the augmentation strategy in [17], the lowest mapping cost is 280, as illustrated in Fig. 39. This difference

doi:10.6342/NTU202501591

suggests that the flexibility of an augmentation strategy plays a crucial role in determining how low the minimum mapping cost can be.

To further understand how the flexibility of augmentation strategies affects the minimum mapping cost, we analyze the solution space size formed by each augmentation strategy. The augmentation strategy in [16] allows for the generation of $(N!)^N = (4!)^4 = 331{,}776$ augmented VNs for the VN depicted in Fig. 1, while [17] generates only $(K)^N = (4)^4 = 256$ augmented VNs. This substantial difference in solution space size indicates that the augmentation strategy in [16] offers significantly greater flexibility compared to [17]. Because of this expanded solution space, the augmentation strategy in [16] can almost identify at least one augmented VN that achieves a lower mapping cost than the minimum attainable under [17].

This is further evident from Fig. 38, where we observe 15 red dots and 11 orange dots, representing different clusters of augmented VNs with mapping costs strictly lower than 280. In total, 415 augmented VNs in the solution space of [16] achieve a mapping cost below 280, an outcome that is unattainable under [17]. Among the 331,776 augmented VNs generated under [16], these 415 instances serve as concrete evidence that a larger solution space increases the likelihood of discovering augmented VNs with lower mapping costs.

Based on this observation, we can reasonably infer that when one augmentation strategy generates a larger solution space than another, it is more likely to include an augmented VN that achieves a lower mapping cost than the minimum mapping cost attainable under the smaller solution space.

doi:10.6342/NTU202501591

This leads to a broader conclusion: to achieve the globally minimal mapping cost, the augmentation strategy must generate the largest solution space. Only when the solution space reaches its maximum can it provide a higher probability than any other augmentation strategy of discovering an augmented VN with a lower minimal mapping cost. A maximal solution space ensures that the globally optimal augmented VN is included, thereby maximizing the likelihood of achieving the lowest possible mapping cost during the embedding phase.

doi:10.6342/NTU202501591

CHAPTER 3
SYSTEM SETTINGS AND ASSUMPTIONS

3.1 Network Model

This section defines the structures and resource attributes of the virtual and substrate networks. It establishes the foundation for understanding the mapping process by specifying the computational and bandwidth requirements of virtual networks and the available capacities of substrate networks.

3.1.1 Virtual Network Model

The virtual network topology V is defined as an 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 $n_v \in N_v$ represents a virtual network node, and each virtual link $l_v \in L_v$ represents a virtual connection between two virtual network nodes. Each virtual node n has a computational demand c_{n_v} , and each virtual link l_v has a bandwidth demand b_{l_v}.

3.1.2 Substrate Network Model

The substrate network topology S is defined as an 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 $n_s \in N_s$ represents a substrate network node in the substrate network, and each substrate link $l_s \in L_s$ represents a physical connection between two substrate network nodes. Each substrate node n_s has an available computational capacity c_{n_s}, and each substrate link l_s has a bandwidth capacity b_{l_s} .

doi:10.6342/NTU202501591

3.2 Resource Constraints and Mapping Limitations

This section explains the constraints and limitations involved in mapping virtual networks onto substrate networks. It outlines the resource requirements during the mapping process and describes the rules ensuring the feasibility and correctness of the mapping.

3.2.1 Resource Constraints during VN to SN Mapping

When mapping a virtual network $G_v = (N_v, L_v)$ onto a substrate network $G_s = (N_s, L_s)$, the resource demands of the virtual network must be satisfied by the available resources of the substrate network. The mapping process is subject to the following constraints:

Node Resource Constraints:

- Each virtual node $n_v \in N_v$ is mapped to a single substrate node $n_s \in N_s$. Let $M_N : N_v \rightarrow N_s$ represent the node mapping function.
- The computational demand c_{n_v} of the virtual node n_v must not exceed the available computational capacity C_{n_s} of the substrate node n_s to which it is mapped:

$$c_{n_v} \leq C_{n_s} \qquad \forall\, n_v \in N_v\,,\ n_s = M_N(n_v)$$

Link Resource Constraints:

- The virtual link $l_v = (n^i_v, n^j_v) \in L_v$ connects two virtual nodes n^i_v and n^j_v, where i, j are indices representing the virtual nodes at the endpoints of the virtual link. This virtual link is mapped to a substrate path P_{l_v}, which consists of a

doi:10.6342/NTU202501591

sequence of substrate links $l_s \in L_s$. Let $M_L : L_v \rightarrow P_{l_v}$ represent the link mapping function.

- The bandwidth demand b_{l_v} of the virtual link $l_v \in L_v$ must not exceed the available bandwidth capacity B_{l_s} of each substrate link l_s in the path P_{l_v}:

$$b_{l_v} \leq B_{l_s} \quad \forall\, l_s \in P_{l_v},\ P_{l_v} = M_L(l_v),\ l_v \in L_v$$

3.2.2 Mapping Limitations

The mapping process is subject to several limitations to ensure correctness and feasibility, as described below:

Distinct Node Mapping Constraint:

- Different virtual nodes from the same virtual network must be mapped to distinct substrate nodes in the substrate network. This condition ensures that in the event of a substrate node failure, only one virtual node from the virtual network is affected. By mapping virtual nodes to distinct substrate nodes, the system avoids the need to prepare excessive backup resources to recover from potential failures affecting multiple virtual nodes.

  This condition can be expressed as:

$$M_N(n^i_v) \neq M_N(n^j_v) \qquad \forall\, n^i_v, n^j_v \in N_v,\ i \neq j$$

  where $M_N : N_v \rightarrow N_s$ represents the node mapping function.

doi:10.6342/NTU202501591

Substrate Path Mapping Constraint:

- The substrate path P_{l_v} mapped for the virtual link $l_v=(n^i_v, n^j_v)$ must connect the substrate nodes $M_N(n^i_v)$ and $M_N(n^j_v)$, where $M_N: N_v \rightarrow N_s$ represent the node mapping function.

3.3 Mapping (Embedding) Cost and Acceptance Ratio Metrics

This section introduces the metrics for evaluating the mapping process. It defines how the total cost of mapping (embedding) a virtual network request is calculated and explains the acceptance ratio metric as a measure of the substrate network's efficiency in accommodating incoming requests.

3.3.1 Mapping (Embedding) Cost Metrics

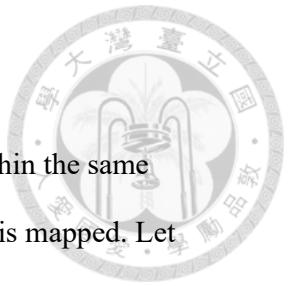
The mapping (embedding) costs of a virtual network request (VNR) reflects the total amount of substrate network resources consumed to map a virtual network $G_v = (N_v, L_v)$. The cost is calculated as the sum of the computational resources allocated to the virtual nodes and the bandwidth resources allocated to the virtual links:

Node Mapping (Embedding) Cost:

- Let c_{n_v} denote the computational demand of a virtual node $n_v \in N_v$ within a virtual network G_v, and $M_N(n_v)$ denote the substrate node to which n_v is mapped. Let w_n represent the cost per unit computational resource for substrate nodes. The node mapping cost is calculated as:

$$\text{Node Mapping Cost} = \sum_{n_v \in N_v} w_n * c_{n_v}$$

doi:10.6342/NTU202501591



Link Mapping (Embedding) Cost:

- Let b_{l_v} denote the bandwidth demand of a virtual link $l_v \in L_v$ within the same virtual network G_v, and P_{l_v} denote the substrate path to which l_v is mapped. Let w_l represent the cost per unit bandwidth resource for substrate links. The link mapping cost is calculated as:

$$\text{Link Mapping Cost} = \sum_{lv \in Lv} \sum_{ls \in P_{lv}} w_l * bl_v$$

Total Mapping (Embedding) Cost:

- The total mapping (embedding) cost for a VNR is the sum of the node mapping cost and the link mapping cost:

Total Mapping Cost = Node Mapping Cost + Link Mapping Cost

doi:10.6342/NTU202501591

3.3.2 Acceptance Ratio Metrics

The acceptance ratio measures the ability of the SN to accommodate incoming VNRs over a given period. This metric is crucial for evaluating the efficiency and scalability of the resource allocation process in the SN.

Definition of Acceptance Ratio:

- The acceptance ratio is defined as the ratio of successfully mapped VNRs to the total number of VNRs that arrive over a specific time period $\boldsymbol{T}$. Let $\boldsymbol{R_{total}}$ denote the total number of VNRs that arrive, and $\boldsymbol{R_{accepted}}$ denote the number of successfully mapped VNRs. The acceptance ratio is calculated as:

$$\text{Acceptance Ratio} = \frac{\boldsymbol{R_{accepted}}}{\boldsymbol{R_{total}}}$$

doi:10.6342/NTU202501591

CHAPTER 4
PROPOSED APPROACH

This chapter presents the proposed approaches to address the challenges of SVNE under single facility node failure scenarios. The goal is to efficiently identify augmented VN with minimal resource increments and map it onto SN with the lowest embedding costs. Section 4.1 identifies the augmentation strategies with the largest solution space and assesses its relative advantage.

Section 4.2 introduces the design of the augmented VN generation algorithm, which is divided into three subsections: Section 4.2.1 identifies the key design principles required to minimize resource increments when generating augmented VN, while Section 4.2.2 presents the proposed Nest-Base heuristic algorithm, which applies these principles to quickly generate the minimum-increment augmented VN , and Section 4.2.3 provides the pseudo code of the Nest-Base algorithm along with a detailed explanation.

Section 4.3 presents the design of the mapping algorithm and is divided into two subsections: Section 4.3.1 describes the proposed mapping algorithm, ensuring that the optimal augmented VN among the candidate solutions obtained in the augmentation phase leads to the lowest mapping (embedding) cost during the mapping (embedding) phase, while Section 4.3.2 presents the pseudo code of the mapping algorithm with a corresponding explanation.

doi:10.6342/NTU202501591

4.1 Optimal Augmented VN Strategy

Following the conclusion in Section 2.3.2, which establishes that an augmentation strategy with a larger solution space increases the likelihood of including an augmented VN with a lower mapping cost, this section aims to identify the strategy that provides the largest solution space. Therefore, determining the augmentation strategy with the largest solution space ensures the highest probability of achieving a global optimum.

However, identifying the augmentation strategy with the largest solution space requires first enumerating all possible augmentation strategies. To achieve this, we must first establish components for defining augmentation strategies.

Through analyzing the augmentation strategies proposed in [16] and [17], we identify four fundamental binary components—backup node quantity, migration rules, link restoration mechanisms, and placement constraints of backup nodes—serve as the core framework for constructing augmented VN. These components are defined as follows:

Backup Node Quantity: This specifies whether a single backup node (S) is used to protect all primary virtual nodes or multiple backup nodes (M) are deployed to provide greater flexibility.

Migration Rules: This defines whether migration is restricted (R), allowing only affected virtual nodes to relocate, or unrestricted (U), permitting all virtual nodes to migrate.

doi:10.6342/NTU202501591

Link Restoration Mechanisms: This specifies whether link restoration is achieved via direct link (D) or path-based routing (P) through intermediate nodes.

Placement Constraints of Backup Nodes: This determines whether backup nodes must be placed on virtual link (L) or are prohibited from being placed on them (N). It is worth noting that, if a backup node is not placed on a virtual link but the link restoration still relies on paths through existing virtual links, then to prevent the backup node from becoming completely isolated from the VN, it is allowed to connect to only one virtual node.

Under the framework established by the four fundamental binary components—backup node quantity, migration rules, link restoration mechanisms, and placement constraints of backup nodes—we systematically derive all 2*2*2*2=16 potential augmentation strategies. Each strategy represents a unique combination of these components, resulting in a distinct strategy for constructing augmented VNs. For clarity, we denote each strategy using a four-letter abbreviation, with each letter corresponding to one of the components.

Table 1 and Table 2 outline these 16 strategies in detail, with each row presenting a unique strategy derived from the possible combinations of these four components. For instance, in the FIP strategy introduced in [16], augmenting VN involves using only a single backup node and restricting migration to affected virtual nodes. Backup links are directly established between two virtual nodes, and backup nodes are prohibited from being placed on virtual link. Based on these characteristics, FIP corresponds to SRDN (No. 1). Similarly, the FDP strategy in [16] extends the flexibility by

doi:10.6342/NTU202501591

permitting all virtual nodes to migrate, while retaining the same conditions for backup nodes, link restoration, and placement constraints of backup nodes as FIP. As a result, FDP corresponds to SUDN (No. 5).

In contrast, the augmentation strategy described in [17] supports multiple backup nodes and restricts migration to affected virtual nodes. Backup links are established by routing through existing virtual links that form a path connecting virtual nodes, and backup nodes are required to be placed on virtual links. Consequently, this approach aligns with MRPL (No. 12).

No.	Strategy	Backup node quantity	Migration rule	Link restoration mechanism	Backup Node Placement
1	SRDN (FIP in [16])	1 (Single)	R (Restricted)	D (Direct)	N (Not on Link)
2	SRDL	1 (Single)	R (Restricted)	D (Direct)	L (On Link)
3	SRPN	1 (Single)	R (Restricted)	P (Path-based)	N (Not on Link)
4	SRPL	1 (Single)	R (Restricted)	P (Path-based)	L (On Link)
5	SUDN (FDP in [16])	1 (Single)	U (Unrestricted)	D (Direct)	N (Not on Link)
6	SUDL	1 (Single)	U (Unrestricted)	D (Direct)	L (On Link)
7	SUPN	1 (Single)	U (Unrestricted)	P (Path-based)	N (Not on Link)
8	SUPL	1 (Single)	U (Unrestricted)	P (Path-based)	L (On Link)

Table 1. First half of 16 augmented VN strategies

doi:10.6342/NTU202501591

No.	Strategy	Backup node quantity	Migration rule	Link restoration mechanism	Backup Node Placement
9	MRDN	1+ (Multiple)	R (Restricted)	D (Direct)	N (Not on Link)
10	MRDL	1+ (Multiple)	R (Restricted)	D (Direct)	L (On Link)
11	MRPN	1+ (Multiple)	R (Restricted)	P (Path-based)	N (Not on Link)
12	MRPL [17]	1+ (Multiple)	R (Restricted)	P (Path-based)	L (On Link)
13	MUDN	1+ (Multiple)	U (Unrestricted)	D (Direct)	N (Not on Link)
14	MUDL	1+ (Multiple)	U (Unrestricted)	D (Direct)	L (On Link)
15	MUPN	1+ (Multiple)	U (Unrestricted)	P (Path-based)	N (Not on Link)
16	MUPL	1+ (Multiple)	U (Unrestricted)	P (Path-based)	L (On Link)

Table 2. Second half of 16 augmented VN strategies

However, not all these strategies are equally efficient in constructing augmented VNs. Some strategies result in augmented VNs that are completely dominated by others in terms of backup resource increments, making them redundant for practical purposes. To streamline the analysis and focus on effective augmentation strategies, the following discussion evaluates these 16 approaches to identify and eliminate inefficient strategies. By doing so, we refine the solution space and retain only the resource-efficient strategies for further analysis.

4.1.1 Identifying Inefficient Augmentation Strategies

We observe that some augmentation strategies allowing multiple backup nodes can be replaced by more efficient strategies using a single backup node. Specifically, among strategies that enforce the restriction of not placing backup nodes on virtual links, all approaches utilizing multiple backup nodes can be substituted by a single backup node method with the same restriction.

The rationale behind this substitution lies in the inefficient of multiple backup nodes under the constraint of not allowing backup nodes to be placed on virtual links. When

doi:10.6342/NTU202501591

a virtual node migrates to a second backup node, additional backup links and bandwidth must be provisioned to establish connections with relevant nodes. In contrast, in a single backup node configuration, the pre-established backup resources of the existing backup node and unaffected virtual nodes can be reused or shared. This reuse reduces the need for new resource allocations. Thus, introducing additional backup nodes under this constraint increases resource usage unnecessarily, as the single backup node configuration is already sufficient to ensure the survivability of the augmented VN.

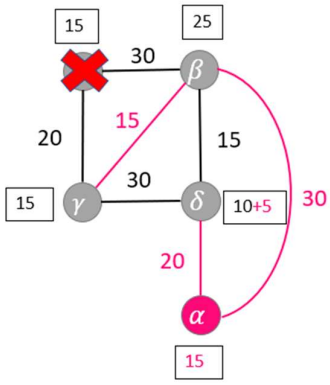


Fig.40 SUDN (No. 5)

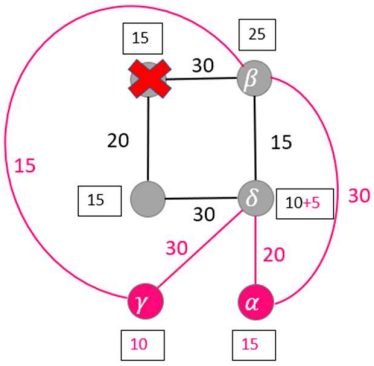


Fig.41 MUDN (No. 13)

Using the VN in Fig. 1 as an example, under the SUDN (No. 5) strategy illustrated in Fig. 40, when the virtual node α is affected and migrates to the backup node, while γ and δ swap positions, 65 units of backup bandwidth and 20 units of backup computational resources are required to restore its functionality. In contrast, Fig. 41 depicts a scenario under the MUDN (No. 13) strategy, where the virtual node γ migrates to a second backup node instead of utilizing the existing virtual node. This configuration requires an additional 30 units of backup bandwidth and 10 units of backup computational resources to establish connections with δ, leading to a higher overall backup resource consumption. This demonstrates that introducing a second

doi:10.6342/NTU202501591

backup node unnecessarily increases the backup resource increment. Since the single backup node design is already sufficient to ensure the survivability of the augmented VN, the use of a second backup node under these constraints is both redundant and inefficient. Similarly, the MRDN (No. 9), MRPN (No. 11), and MUPN (No. 15) strategies are all less efficient compared to the SRDN (No. 1), SRPN (No. 3), and SUPN (No. 7) strategies, respectively.

However, beyond this, strategies that allow multiple backup nodes while enforcing the placement of backup nodes on virtual links and relying on direct links for link restoration can also be replaced by a single backup node method under the same constraints.

The rationale behind this substitution lies in the redundancy of multiple backup nodes under the constraint of enforcing the placement of backup nodes on virtual links and restricting link restoration to direct links. In this scenario, the primary advantage of multiple backup nodes—allowing virtual nodes to migrate to backup nodes closer to the target virtual node for service restoration—becomes irrelevant. This is because, with direct link restoration, all backup links are fixed at 1 hop in length. Consequently, in the augmented virtual network, there is no effective distance difference between virtual nodes for restoring services. As a result, multiple backup nodes fail to reduce the increment of backup resources and instead lead to inefficient and unnecessary resource allocation.

doi:10.6342/NTU202501591

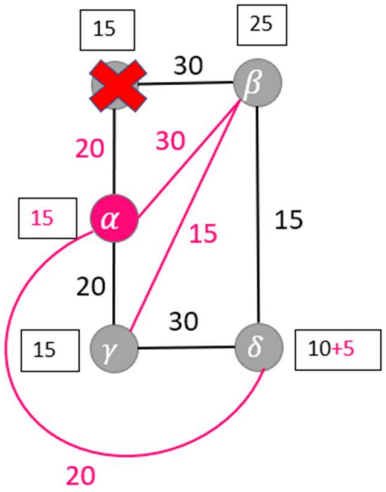


Fig.42 SUDL (No. 6)

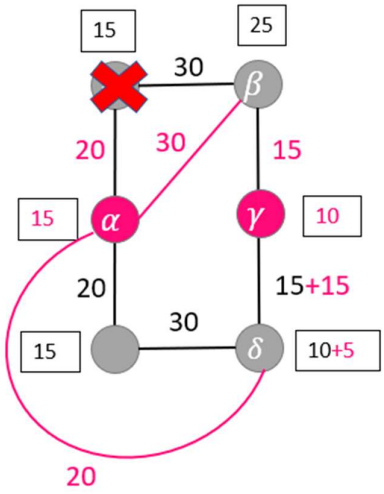


Fig.43 MUDL (No. 14)

Using the VN in Fig. 1 as an example, under the SUDL (No. 6) strategy illustrated in Fig. 42, when the virtual node α is affected and migrates to the backup node, 20 units of backup bandwidth need to be added between the backup node and the affected node to ensure normal transmission services before any failure occurs. Additionally, γ and δ swap positions, requiring a total of 85 units of backup bandwidth and 20 units of backup computational resources to restore functionality. In contrast, Fig. 43 illustrates a different scenario under the MUDL (No. 14) strategy, where the node γ migrates to a second backup node instead of utilizing the existing virtual node. This configuration requires an additional 15 units of backup bandwidth between γ and β to ensure normal transmission services before any failure occurs, as well as 15 additional units of backup bandwidth to meet the bandwidth demands between γ and δ. Furthermore, 10 units of backup computational resources are required. This results in higher overall backup resource consumption. This example demonstrates that employing multiple backup nodes under such constraints is redundant and inefficient. Similarly, the MRDL (No. 10) strategy is less efficient compared to the SRDL (No. 2) strategy.

doi:10.6342/NTU202501591

4.1.2 Analyzing the Solution Space of Effective Strategies

Having filtered out redundant strategies in Section 4.1.1, this section delves into the analysis of the solution space for the 10 remaining effective augmentation strategies. These include 8 strategies that allow only a single backup node (No. 1 to No. 8 in Table 1, or SXXX, where X serves as a placeholder for either binary choice in each category) and 2 strategies that permit multiple backup nodes (No. 12 and No. 16 in Table 2, or MXPL).

To systematically examine the solution space, we first analyze the single backup node strategies. We further group together strategies that share the same components except for their link restoration mechanisms (SRXN, SRXL, SUXN, SUXL). In each of these 4 subcategories, the size of the solution space of any strategy will be the same. The reason is that under direct link conditions (D), there is only one recovery link between nodes. Similarly, under path-based conditions (P), the recovery path is uniquely determined by a weighted shortest path algorithm.

In some cases, different migration configurations within the same augmentation strategy can result in identical augmented VNs. To distinguish between these cases in the solution space, we define each feasible solution as an augmented VN characterized by its Migration Set and the corresponding resource increment. Within the solution space, each feasible solution is defined by the following components:

doi:10.6342/NTU202501591

Migration Set: The migration of virtual nodes under each failure scenario, specifying where the affected node migrates and how the unaffected nodes adjust within the augmented VN.

Increments: The total resource increment, representing the additional backup resources required for the augmented VN.

For instance, as discussed in Section 2.2.2.2 regarding the FIP strategy, considering the VN in Fig. 1 and applying the SRDN (No. 1) strategy, the resulting augmented VN aligns with the FIP strategy proposed in [16], with a backup resource increment of 145. The corresponding feasible solution can be expressed as {**Migration Set**, 145}. Here, **Migration Set** is defined as:

$$\textbf{Migration Set} = \{\alpha{:}(\pi, \beta, \gamma, \delta),$$
$$\beta{:}(\alpha, \pi, \gamma, \delta),$$
$$\gamma{:}(\alpha, \beta, \pi, \delta),$$
$$\delta{:}(\alpha, \beta, \gamma, \pi)\}$$

The Migration Set, represented as {α:(π, β, γ, δ), β:(α, π, γ, δ), γ:(α, β, π, δ), δ:(α, β, γ, π)}, defines the reallocation of virtual nodes under each failure scenario. Each entry, such as α:(π, β, γ, δ), specifies the node arrangement when α is affected. The positions within the parentheses indicate the locations of α, β, γ, and δ during this failure scenario. Here, π represents a backup node available for migration, and in this case, the affected node α migrates to π, while the remaining nodes (β, γ, and δ) remain in their original positions.

doi:10.6342/NTU202501591

Similarly, considering the SUDN (No. 5) strategy under the FDP strategy proposed in [16], the heuristic-derived augmented VN can be represented as {**Migration Set**, 160}. Here, **Migration Set** is defined as:

$$\textbf{Migration Set} = \{\alpha:(\pi, \beta, \gamma, \delta),$$
$$\beta:(\pi, \alpha, \gamma, \delta),$$
$$\gamma:(\alpha, \beta, \pi, \delta),$$
$$\delta:(\alpha, \beta, \gamma, \pi)\}$$

For example, when β is affected, the configuration is β:(π, α, γ, δ), where β migrates to the vacant node originally occupied by α, and α relocates to the backup node π. The other nodes, γ and δ, remain unchanged. In other failure scenarios, only the affected node migrates to π, while all other nodes stay in their original positions.

Solution Space of SRDN (No. 1) and SRPN (No. 3)

The strategies SRDN (No. 1) and SRPN (No. 3) are both characterized by the restriction that only the affected virtual node can migrate to the backup node (R), and only a single backup node is used (S). Under these constraints, for any given failure scenario, there is only one possible migration arrangement: the affected virtual node migrates to the backup node while other virtual nodes remain unchanged. This limitation results in a unique arrangement for every failure scenario. Consequently, the solution space for both SRDN and SRPN consists of a single augmented VN, making the size of their solution space equal to 1.

doi:10.6342/NTU202501591

Solution Space for SRDL (No. 2) and SRPL (No. 4)

The SRDL (No. 2) and SRPL (No. 4) strategies share similar characteristics with SRDN (No. 1) and SRPN (No. 3), except for one key distinction: the backup node is required to be placed on a virtual link.

For a VN with K virtual links, the backup node in these strategies can be placed in K possible locations. However, once the location of the backup node is determined for an augmented VN, this placement of backup node remains fixed across all failure scenarios. Consequently, for each failure scenario, only the affected virtual node migrates to the pre-determined backup node, while the unaffected nodes remain unchanged. As a result, the total number of augmented VNs, and thus the size of the solution space for SRDL (No. 2) and SRPL (No. 4), is K. This reflects the K distinct ways the backup node can be placed on the virtual links, with each placement of backup node corresponding to a unique augmented VN.

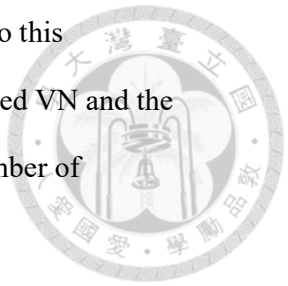
Solution Space for SUDN (No. 5) and SUPN (No. 7)

The SUDN (No. 5) and SUPN (No. 7) strategies differ from SRDN (No. 1) and SRPN (No. 3) in their migration rules. While SRDN and SRPN restrict migration to the affected virtual node only, SUDN and SUPN allow both affected and unaffected virtual nodes to migrate.

For a VN with N virtual nodes, each failure scenario permits N! distinct migration arrangements, where N! is calculated as the number of all possible arrangements of N virtual nodes. It is important to note that the backup resources required for each failure scenario must be pre-allocated and shared across all failure scenarios. Consequently, the total backup resource increment for an augmented VN is the

doi:10.6342/NTU202501591

cumulative sum of the resources required for all failure scenarios. Due to this relationship between the total backup resource increment of an augmented VN and the backup resource increment for individual failure scenarios, the total number of feasible solutions under SUDN (No. 5) and SUPN (No. 7) is $(N!)^N$.



Solution Space for SUDL (No. 6) and SUPL (No. 8)

The SUDL (No. 6) and SUPL (No. 8) strategies differ from SUDN (No. 5) and SUPN (No. 7) only in the placement constraints for backup node. Specifically, SUDL and SUPL require the backup node to be placed on a virtual link.

In SUDN (No. 5) and SUPN (No. 7), the number of feasible solutions is $(N!)^N$, where N! represents the number of migration arrangements for each of the N failure scenarios.

However, unlike SUDN (No. 5) and SUPN (No. 7), SUDL (No. 6) and SUPL (No. 8) allow K possible placement options for the backup node on virtual links. As a result, the solution space for SUDL (No. 6) and SUPL (No. 8) expands to $K*(N!)^N$, meaning that the solution space is influenced not only by the migration arrangements and the number of failure scenarios but also by the number of potential backup node placements on virtual links.

Solution Space for MRPL (No. 12)

The MRPL (No. 12) strategy allows multiple backup nodes to be placed on virtual links, but only the affected virtual node is permitted to migrate and utilize a backup node in the event of a failure. For each failure scenario, the affected virtual node can freely select one of the K virtual links to place and use a backup node, enabling

doi:10.6342/NTU202501591

backup resources allocated for one failure scenario to be shared across others. Consequently, the total backup resource increment required for an augmented VN corresponds to the cumulative backup resources needed across all failure scenarios. Given that a VN consists of N virtual nodes, each with K possible placement options for a backup node, the total number of possible augmented VNs under the MRPL (No. 12) strategy is K^N.

Solution Space for MUPL (No. 16)

The MUPL (No. 16) strategy extends the flexibility of node migration compared to the MRPL (No. 12) strategy. While MRPL (No. 12) allows only the affected virtual node to migrate during a failure scenario, MUPL (No. 16) removes this restriction, enabling any virtual node—whether affected or unaffected—to migrate under any failure scenario.

For a VN with N virtual nodes and K virtual links, during each failure scenario, up to N−1 unaffected nodes and K backup nodes can be used for recovery. This results in P_N^{N+K-1} potential migration arrangements for each failure scenario. Additionally, the total backup resource increment required for an augmented VN corresponds to the cumulative backup resources needed across all failure scenarios. Consequently, the solution space size under the MUPL strategy is calculated as $(P_N^{N+K-1})^N$, reflecting the vast number of possible augmented VNs that can be generated using this strategy.

doi:10.6342/NTU202501591

No.	Strategy	Backup node quantity	Migration rule	Link restoration mechanism	Backup Node Placement	Solution space
1	SRDN (FIP in [16])	1 (Single)	R (Restricted)	D (Direct)	N (Not on Link)	1
2	SRDL	1 (Single)	R (Restricted)	D (Direct)	L (On Link)	K
3	SRPN	1 (Single)	R (Restricted)	P (Path-based)	N (Not on Link)	1
4	SRPL	1 (Single)	R (Restricted)	P (Path-based)	L (On Link)	K
5	SUDN (FDP in [16])	1 (Single)	U (Unrestricted)	D (Direct)	N (Not on Link)	$(N!)^N$
6	SUDL	1 (Single)	U (Unrestricted)	D (Direct)	L (On Link)	$K*(N!)^N$
7	SUPN	1 (Single)	U (Unrestricted)	P (Path-based)	N (Not on Link)	$(N!)^N$
8	SUPL	1 (Single)	U (Unrestricted)	P (Path-based)	L (On Link)	$K*(N!)^N$

Table 3. First half of 16 augmented VN strategies with corresponding solution space sizes

No.	Strategy	Backup node quantity	Migration rule	Link restoration mechanism	Backup Node Placement	Solution space
9	MRDN	1+ (Multiple)	R (Restricted)	D (Direct)	N (Not on Link)	don't care
10	MRDL	1+ (Multiple)	R (Restricted)	D (Direct)	L (On Link)	don't care
11	MRPN	1+ (Multiple)	R (Restricted)	P (Path-based)	N (Not on Link)	don't care
12	MRPL [17]	1+ (Multiple)	R (Restricted)	P (Path-based)	L (On Link)	K^N
13	MUDN	1+ (Multiple)	U (Unrestricted)	D (Direct)	N (Not on Link)	don't care
14	MUDL	1+ (Multiple)	U (Unrestricted)	D (Direct)	L (On Link)	don't care
15	MUPN	1+ (Multiple)	U (Unrestricted)	P (Path-based)	N (Not on Link)	don't care
16	MUPL	1+ (Multiple)	U (Unrestricted)	P (Path-based)	L (On Link)	$(P_N^{N+K-1})^N$

Table 4. Second half of 16 augmented VN strategies with corresponding solution space sizes

With the analysis of all augmentation strategies completed, the corresponding solution space sizes for each strategy are summarized in Tables 3 and 4. For certain strategies, the "Solution Space" column is marked as "don't care," indicating that these strategies have been classified as inefficient and are therefore excluded from further consideration.

doi:10.6342/NTU202501591

With all possible augmentation strategies identified and their corresponding solution space sizes computed, the next challenge is to determine which augmentation strategy provides the largest solution space.

However, merely identifying the largest solution space is not sufficient. It is crucial to evaluate how much larger this solution space is compared to all other strategies. If the largest solution space is only marginally larger than all other strategies, its advantage may be negligible. To quantify this relative advantage, we consider the Augmented VN Space, which encompasses the solution spaces of all efficient augmentation strategies. By analyzing the composition of this space, we measure the proportion occupied by the strategy with the largest solution space relative to the Augmented VN Space.

Among the fundamental binary components defining augmentation strategies, only the placement constraints of backup nodes and link restoration mechanisms influence the mutual exclusivity of solution spaces. These components govern whether solution spaces overlap or remain disjoint. This analysis becomes particularly significant for augmentation strategies allowing only a single backup node.

In the case of backup node placement, augmentation strategies categorized as "On Link" (L) and "Not on Link" (N) exhibit mutually exclusive solution spaces with no intersections. Compared to the "Not on Link" (N) method, the "On Link" (L) method places backup nodes on virtual links, which alters the original VN structure by removing the direct connection between two previously adjacent virtual nodes. As a result, the augmented VN structures under the "Not on Link" (N) and "On Link" (L) methods are inherently different, preventing any intersection between their solution

doi:10.6342/NTU202501591

spaces.

Similarly, for link restoration mechanisms, augmentation strategies using direct links (D) as restoration and those using path-based routing (P) also exhibit mutually exclusive solution spaces when N>2. This mutual exclusivity arises because single backup node methods employing path-based routing cannot ensure that all restoration paths in every failure scenario are limited to a single hop. When restoration paths require more than one hop, the structural differences between direct link and path-based routing result in divergent augmented VN configurations, preventing any intersection between their solution spaces.

Under the fundamental binary components governing augmentation strategies, backup node quantity and migration rules primarily influence the size of the solution space. Regarding backup node quantity, allowing multiple backup nodes (M) represents a more relaxed constraint compared to permitting only a single backup node (S). Consequently, under the same migration rule, the solution space of the former fully encompasses that of the latter.

Similarly, for migration rules, allowing unrestricted migration of any virtual node (U) constitutes a less restrictive condition compared to permitting migration only for affected virtual nodes (R). Thus, under the same backup node quantity constraint, the solution space of the former entirely includes the solution space of the latter.

Additionally, if only a single backup node is allowed (S) but the migration rules are more relaxed (U), and conversely, if multiple backup nodes are permitted (M) but the migration rules are stricter (R), their solution spaces may partially overlap.

doi:10.6342/NTU202501591

Specifically, the overlap occurs where only a single backup node is allowed (S) and the migration rules are stricter (R).

From Tables 3 and 4, we observe that the sizes of the solution spaces across different augmentation strategies exhibit a clear hierarchy:

$$1 < K < K^N < (N!)^N < K*(N!)^N < (P_N^{N+K-1})^N$$

The relationship K < N! is inevitable because K, representing the number of virtual links in the VN, falls within the range $N - 1$ to $\frac{N*(N-1)}{2}$ whereas N! grows factorially with the number of virtual nodes N.

For example, consider the VN in Fig. 1, where N=4 and K=4:

- $K^N = 4^4 = 256$
- $(N!)^N = (4!)^4 = 331776$
- $K*(N!)^N = 4*(4!)^4 = 1327104$
- $(P_N^{N+K-1})^N = (P_4^7)^4 = 497871360000$

This numerical example highlights how the solution space size grows exponentially as the augmentation strategies adopt more relaxed constraints.

Building on the discussion above, Fig. 44 summarizes the solution spaces of all augmentation strategies, along with their respective sizes and mutual relationships.

doi:10.6342/NTU202501591

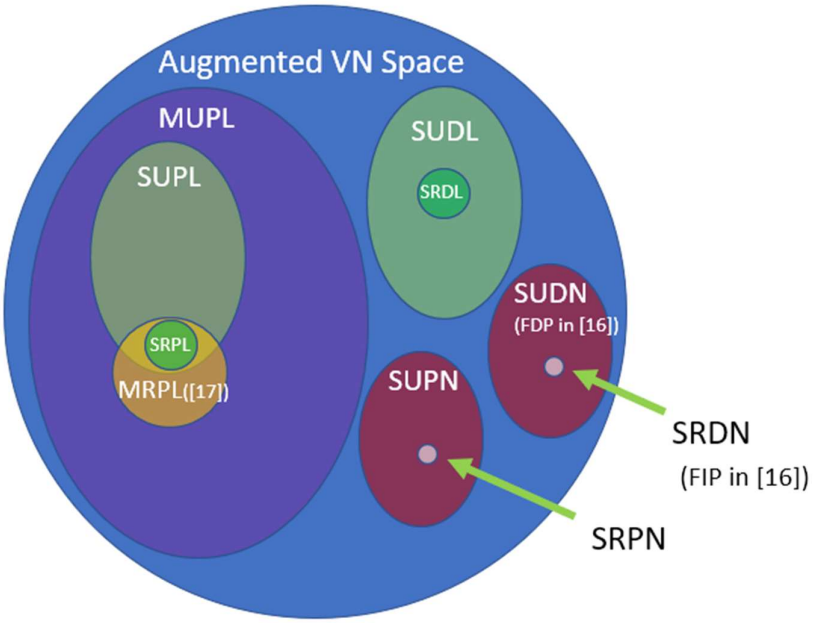


Fig.44 Illustration of solution space sizes and interactions of all augmentation strategies (not to scale)

From Fig. 44, it is evident that the Augmented VN Space is composed of the mutually exclusive solution spaces of MUPL, SUDL, SUDN, and SUPN. This relationship can be expressed as:

$$\text{Augmented VN Space} = \text{MUPL} \cup \text{SUDL} \cup \text{SUDN} \cup \text{SUPN}.$$

Based on the previous analysis, MUPL represents the largest subspace within the Augmented VN Space. As a result, the globally optimal augmented VN is most likely to be found within the solution space of the MUPL strategy.

Although the augmentation strategy with the largest solution space has the highest probability of containing the globally optimal augmented VN, the exact distribution of the globally optimal augmented VN within the Augmented VN Space remains

doi:10.6342/NTU202501591

unknown. This uncertainty arises because the mapping cost of an augmented VN is influenced by multiple factors related to the SN, including the number of substrate nodes and links, the SN's topology, and the available computational and bandwidth resources of each substrate node and link. Given these complexities, we assume that the globally optimal augmented VN is uniformly distributed across the Augmented VN Space.

The probability of including the Globally Optimal Augmented VN in MUPL's solution space can be approximated as the ratio of MUPL's solution space size to the total size of the Augmented VN Space:

$$\frac{|\text{MUPL}|}{|\text{Augmented VN Space}|} = \frac{|\text{MUPL}|}{|\text{MUPL}| + |\text{SUDL}| + |\text{SUDN}| + |\text{SUPN}|}$$

For example, consider the VN in Fig. 1, where N=4 and K=4: the probability of MUPL's solution space containing the Globally Optimal Augmented VN can be expressed as:

$$\frac{497871360000}{497871360000 + 1327104 + 331776 + 331776} = 99.999600168\%$$

Such a high probability demonstrates that using the MUPL strategy to augment VN is almost certain to include the Globally Optimal Augmented VN.

This highlights the significance of MUPL's expansive solution space in increasing the likelihood of achieving the globally optimal solution, emphasizing its critical role

doi:10.6342/NTU202501591

among the available augmentation strategies. Notably, MUPL's solution space is not only the largest among all augmentation strategies but also dominates nearly the entire Augmented VN Space. As demonstrated in the example, MUPL's solution space accounts for an overwhelming 99.999600168% of the total Augmented VN Space, indicating that the probability of it containing the globally optimal augmented VN is extremely high. This overwhelming proportion underscores that using the MUPL strategy almost guarantees the inclusion of the globally optimal augmented VN, further solidifying its importance in augmentation-based optimization.

4.2 Design for Augmented VN Generation Algorithm

4.2.1 Design Principles for Minimum-Increment Augmented VN

From the discussion in Section 4.1, the MUPL strategy provides the largest solution space among all augmented VN strategies and is the most likely to include the globally optimal augmented VN. Therefore, the MUPL strategy is adopted for augmented VN generation. However, the size of the solution space under the MUPL strategy grows exponentially with the VN size, making exhaustive enumeration computationally infeasible. To address this challenge, a method is necessary to efficiently identify the minimum-increment augmented VN within the MUPL solution space.

In our method, each virtual node’s position is regarded as a “Nest”, representing a potential destination for future migrations. When a facility node failure occurs, the virtual node migration process starts from the affected virtual node and relocates it to the nearest available Nest.

doi:10.6342/NTU202501591

Take Fig. 19 and Fig. 20 as an example again. Node π is the Nest of nodes α and β whey they are affected.

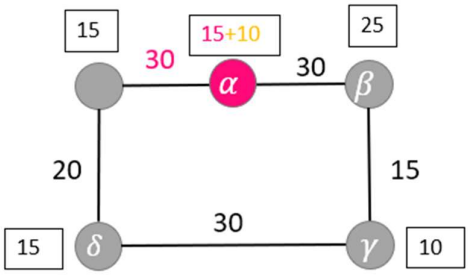


Fig.45 α migrates to backup node

Before further considering the case where virtual node δ is affected, it can be observed that even when no virtual node is affected, node α can still proactively migrate to the backup node and continue to function normally as illustrated in Fig. 45. This is because the backup node has already been verified to successfully restore α's service when α is affected. Therefore, the proactive migration of α to the backup node can be performed safely even in the absence of failures.

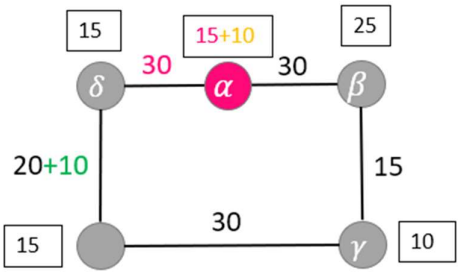


Fig.46 δ occupies α's vacant node

As a result of α's migration to the backup node, α's original Nest becomes vacant. When virtual node δ is affected, it can migrate into this vacant Nest, allowing δ to

doi:10.6342/NTU202501591

fully reuse the released computational resources. Furthermore, δ and α can fully share the bandwidth originally reserved for the virtual link between them. Since δ has migrated to a new Nest, an additional 10 units of backup needs to be provisioned to fully restore the link bandwidth required between δ and γ, as shown in Fig. 46.

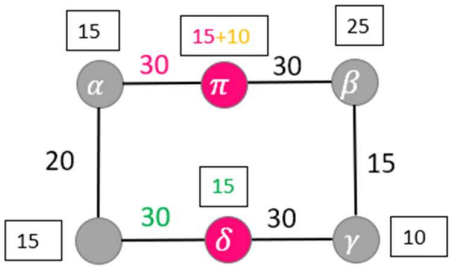


Fig.47 Adding a backup node to protect δ

In addition to migrating δ into the vacant Nest, as illustrated in Fig. 46, another option to protect δ is to prepare a new backup node specifically for δ. Similar to the protection of α, where the backup node was placed along the virtual link with the highest bandwidth demand, the new backup node for δ can be placed along the virtual link between δ and γ, which has the highest bandwidth demand among δ's adjacent links. δ can then migrate into this newly prepared backup node, as shown in Fig. 47.

For δ, two protection options are available: migrating into a vacant Nest, which requires a resource increment of 10 units, or preparing a new backup node, which requires a resource increment of 45 units. Following the minimum-increment principle, the migration option is selected, as shown in Fig. 46. Similarly, for α and β, it is also possible to prepare new backup nodes along their highest-bandwidth virtual links. However, since these links already have backup nodes in place, the migration option and the new backup node preparation option become equivalent for α and β.

doi:10.6342/NTU202501591

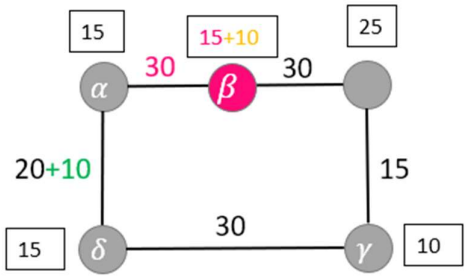


Fig.48 β migrates to backup node

Next, for the protection of γ, two options are also available: the migration option and the backup node preparation option. Starting with the migration option, it can be observed that γ's neighboring nodes, β and δ, have already been protected. Furthermore, when no failure occurs, β can proactively migrate to its designated backup node—previously prepared during β's protection process—and continue to function normally, as shown in Fig. 48. Similarly, δ can also proactively migrate, following the process shown in Fig. 46, and continue to operate without interruption.

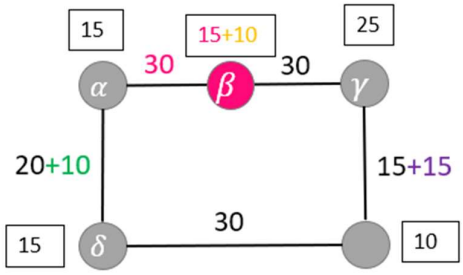


Fig.49 γ occupies β's vacant node

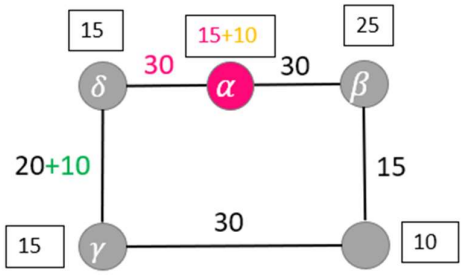


Fig.50 γ occupies δ's vacant node

When γ is affected, it can migrate into the vacant Nest originally occupied by β, as shown in Fig. 49, or into the Nest originally occupied by δ, as shown in Fig. 50. The migration option shown in Fig. 49 requires a backup resource increment of 15 units,

doi:10.6342/NTU202501591

while the migration option shown in Fig. 50 requires no additional backup resources.

On the other hand, the backup node preparation option would place a new backup node along the virtual link between γ and δ, requiring a backup resource increment of 40 units. Therefore, the migration option shown in Fig. 50 is ultimately selected as it incurs the lowest resource increment.

Finally, the augmented VN designed under the proposed method requires only 25 units of backup computation resources and 40 units of backup bandwidth resources, resulting in a total backup resource increment of 65 units. This is lower compared to the FIP and FDP methods in [16], which require 145 and 160 units, respectively, and the ProRed method in [17], which requires 100 units. This demonstrates the effectiveness of the proposed method in achieving augmented VNs with minimal backup resource increments.

4.2.2 Nest-Base Algorithm for Minimum-Increment Augmented VN Generation

Building upon the proposed method identified in Section 4.2.1, the proposed Nest-Base algorithm is designed to quickly generate the minimum-increment augmented VN under the MUPL strategy. By leveraging resource sharing and reuse while minimizing backup resource increments, this algorithm provides an efficient solution for navigating the expansive solution space of the MUPL strategy.

doi:10.6342/NTU202501591

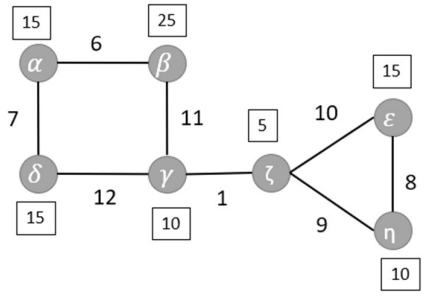


Fig.51 7-nodes VN

VN node	Degree	Sum of adjacent link bandwidths
α	2	6+7=13
β	2	6+11=17
γ	**3**	**1+11+12=24**
δ	2	7+12=19
ε	2	8+10=18
ζ	**3**	1+9+10=20
η	2	8+9=17

Fig.52 Virtual node metric values

To illustrate the operational process of the Nest-Base algorithm, we consider the VN depicted in Fig. 51. In the augmentation process using the Nest-Base algorithm, only one virtual node is considered as affected and requires protection at a time. The process begins by selecting the first virtual node to be protected, prioritizing the node with the highest degree. If multiple nodes share the highest degree, the selection is further refined by choosing the node with the highest total sum of adjacent link bandwidths. As shown in Fig. 52, among all virtual nodes, γ and ζ both have the highest degree of three. To break the tie, the node with the highest sum of adjacent link bandwidths is chosen. γ has a total adjacent bandwidth of 24, while ζ has 20. Consequently, γ is selected as the first node to be protected.

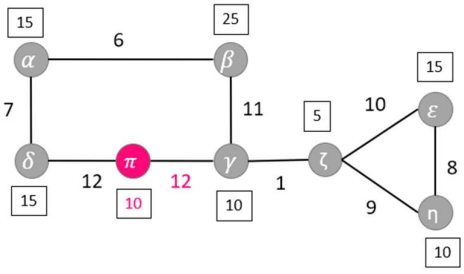


Fig.53 Adding first backup node

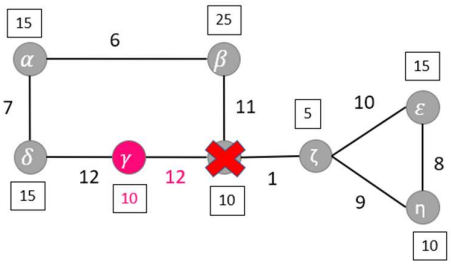


Fig.54 Protection for γ when affected

doi:10.6342/NTU202501591

To protect γ, a backup node is deployed based on the bandwidth demands of its adjacent virtual links. As shown in Fig. 51, γ is connected to δ, β, and ζ, with respective bandwidth demands of 12, 11, and 1. Since the link between γ and δ has the highest bandwidth demand, the backup node is placed on this link to maximize resource utilization, as shown in Fig. 53. When γ is affected, it is protected by migrating to the backup node. The algorithm verifies whether service continuity can be maintained under the current resource conditions. As illustrated in Fig. 54, the existing bandwidth resources can be fully reused to satisfy the bandwidth demands. Therefore, no additional backup bandwidth is required.

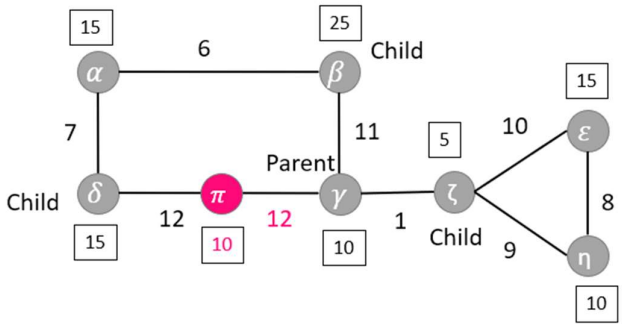


Fig.55 Defining γ's adjacent nodes as child nodes

After protecting the first virtual node (γ), it is designated as the parent node, while all its adjacent virtual nodes are labeled as child nodes, which are the next nodes to be protected, as illustrated in Fig. 55. When a child node is affected, if there is no backup node between the child and the parent, the child node can migrate directly to the parent's location, allowing for resource reuse. Alternatively, if a backup node is already placed on the link connecting the child node and the parent node, the child node will instead migrate to this backup node, also ensuring the efficient reuse of resources.

doi:10.6342/NTU202501591

When protecting a child node, two options are considered to ensure survivability while minimizing backup resource increments. The first option migrates the affected child node to the parent's location if no backup node exists between them; otherwise, the child migrates to the backup node on the link. The second option introduces a new backup node on the link with the highest bandwidth demand, allowing the child node to migrate there. The algorithm evaluates both options and selects the one with the lowest resource increment.

The child nodes are protected in descending order based on their bandwidth demand with the parent node. As shown in Fig. 55, γ is connected to δ, β, and ζ, with respective bandwidth demands of 12, 11, and 1. Therefore, the protection order for the child nodes is δ, β, and ζ.

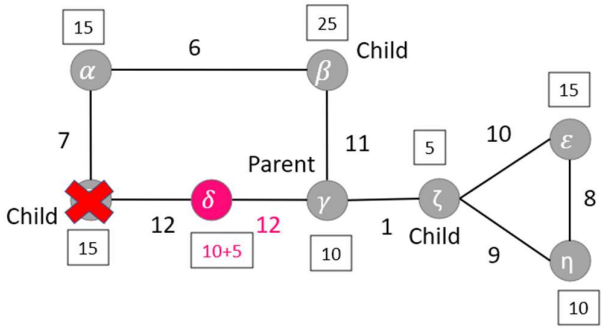


Fig.56 Both options yield the same result for δ's protection

When δ is affected, it has two possible protection options. It can either migrate to the backup node already placed on the link between δ and the parent node γ, as shown in Fig. 56, or a new backup node can be introduced on the link with the highest bandwidth among δ's adjacent links. However, since the highest-bandwidth link is precisely the one connecting δ to γ, and a backup node already exists there, both

doi:10.6342/NTU202501591

options lead to the same result, as illustrated in Fig. 56.

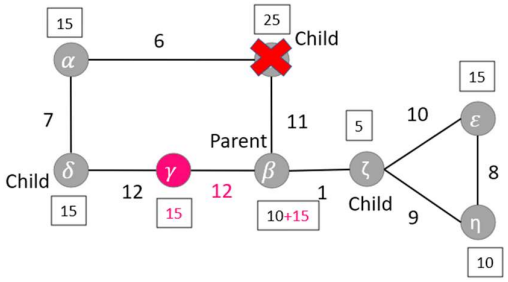


Fig.57 First option for protecting β

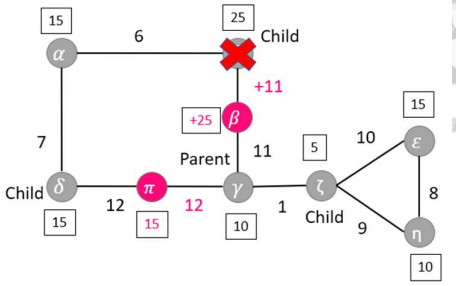


Fig.58 Second option for protecting β

After protecting δ, the next child node to be protected is β. Two possible options are evaluated. The first option, illustrated in Fig. 57, involves migrating β to the parent node γ. Since γ's position is now occupied, γ subsequently migrates to its designated backup node, as it would when affected by a failure in the SN. The second option, illustrated in Fig. 58, introduces a new backup node on the virtual link between γ and β to provide direct protection for β.

A comparison of backup resource increments shows that the first option requires an additional 15 backup resource units, whereas the second option requires 36 units. Since the first option incurs a lower backup resource increment, it is selected for the protection of β, as shown in Fig. 57.

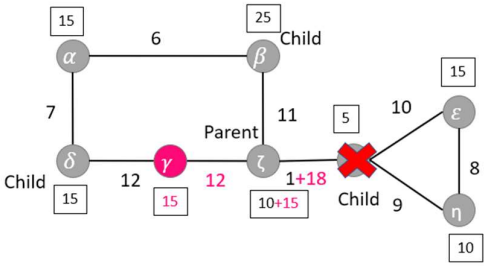


Fig.59 First option for protecting ζ

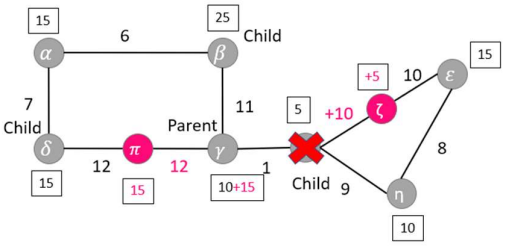


Fig.60 Second option for protecting ζ

doi:10.6342/NTU202501591

After protecting β, the next child node to be protected is ζ, which also has two possible protection options. The first option, illustrated in Fig. 59, involves migrating ζ to the parent node γ. Since γ's position is now occupied, it must migrate to its designated backup node, just as it would in the event of an impact. The second option, illustrated in Fig. 60, introduces a new backup node on the virtual link between ζ and ε, allowing ζ to migrate to this backup node instead. The backup resource increments for the two options are 18 and 15, respectively. Based solely on backup resource increments, the second option is preferable due to its lower resource consumption.

To maintain a linear relationship between the augmented VN's resource increment and its final embedding cost with a slope of 1, it is essential to maximize the likelihood that all virtual links can be mapped using 1-hop paths. This ensures that the bandwidth cost in the substrate network increases exactly in proportion to the bandwidth demand of each virtual link in the augmented VN.

Compared to the first option, which avoids introducing new virtual links and therefore has a higher chance of achieving 1-hop mappings, the second option introduces an additional backup virtual link, increasing the number of virtual links and reducing the likelihood that all virtual links can be mapped using 1-hop paths, making it more challenging to preserve the desired linear relationship.

Although the second option results in more virtual links, making it harder to achieve 1-hop mappings compared to the first option, it also requires a smaller resource increment than the first option. This implies that if both options could successfully achieve 1-hop mappings, the second option would incur a lower final embedding cost than the first option. Therefore, to ensure that no potentially superior configuration is

doi:10.6342/NTU202501591

prematurely discarded, the algorithm retains all optimal augmented VN candidates across different backup node quantities, which directly determine the number of virtual links. By preserving this complete set of candidates, the mapping phase can comprehensively evaluate all options and ultimately select the candidate with the lowest embedding cost.

At this stage, the augmentation process branches into two separate paths, referred to as Branch 1 and Branch 2, based on the protection options applied to ζ, as shown in Fig. 59 and Fig. 60. These branches will be carried forward for further evaluation.

Branch 1

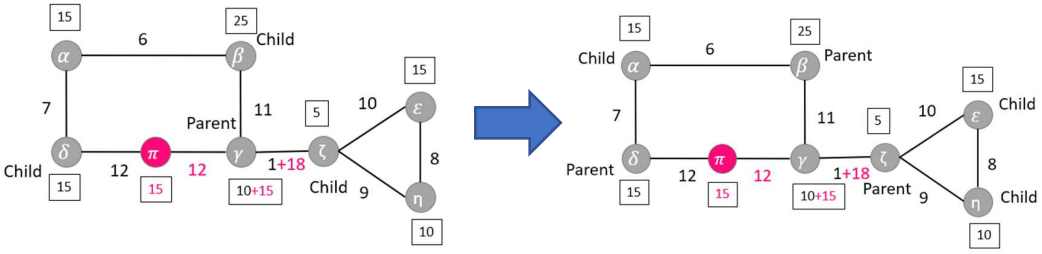


Fig.61 Parent-child node transition

With all child nodes surrounding the parent node γ now protected, they are promoted to parent nodes, while their adjacent, yet unprotected virtual nodes are designated as new child nodes, as illustrated in Fig. 61. When these new child nodes (ε,η,α) are affected, they migrate to the nearest parent node's position, maximizing resource sharing and reuse.

doi:10.6342/NTU202501591

Next, the child nodes are sequentially protected based on their maximum bandwidth demand with their respective parent nodes. To determine the protection order, these child nodes are sorted in descending order of their bandwidth demands. As shown in Fig. 59, ε has the highest bandwidth demand (10), followed by η (9), and finally α (7). Therefore, the protection sequence follows this order: first ε, then η, and finally α.

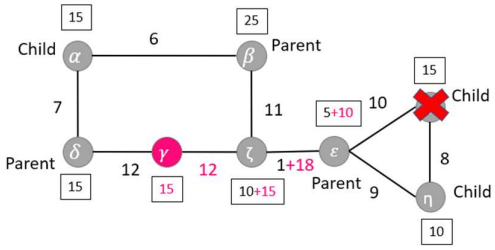


Fig.62 First option for protecting ε

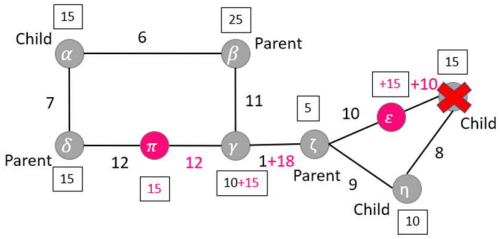


Fig.63 Second option for protecting ε

For the protection of ε, two options are considered. The first option involves migrating ε to its parent node ζ. Since ζ's position is now occupied, ζ subsequently migrates to the position of γ, as it would in the event of its failure. Likewise, γ, now displaced, migrates to its designated backup node, as illustrated in Fig. 62.

The second option introduces a new backup node on the virtual link between ε and ζ to provide protection for ε, as shown in Fig. 63. The backup resource increments required for these options are compared, with the first option requiring 10 units, while the second option requires 25 units. Since the first option results in a lower backup resource increment, it is selected, as illustrated in Fig. 62.

doi:10.6342/NTU202501591

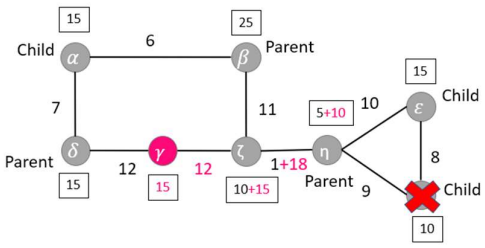


Fig.64 First option for protecting η: occupying ζ's vacant node

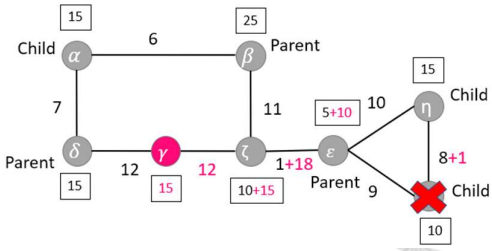


Fig.65 First option for protecting η: occupying ε's vacant node

After protecting ε, the next step is to ensure the survivability of η. Two options are evaluated: migration through node occupation and introducing a new backup node. In this case, η is adjacent to its parent node ζ and its sibling node ε. A key observation is that both ζ and ε have already been protected, meaning that, under normal operational conditions, either node can relocate without disrupting service. This allows η to migrate either to its parent node ζ or to the previously protected sibling node ε.

In the migration option, the first approach involves η relocating to its parent node ζ, following the same migration chain as in previous cases, as illustrated in Fig. 64. The second approach involves η occupying the position of ε, which also triggers a sequence of node migrations, as shown in Fig. 65. Comparing backup resource increments, the first approach requires no additional backup resources, whereas the second approach incurs an increment of 1 unit. As a result, the first approach is selected, as illustrated in Fig. 65, ensuring efficient resource utilization.

The backup node option, which introduces a new backup node, is not further considered, as it inherently increases the backup resource increment.

doi:10.6342/NTU202501591

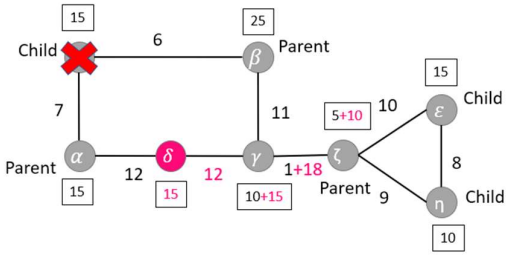


Fig.66 First option for protecting α: occupying δ's vacant node

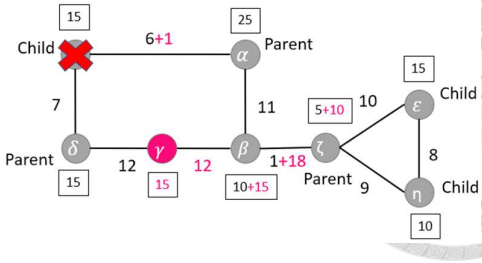


Fig.67 First option for protecting α: occupying β's vacant node

After protecting η, the next step is to ensure the survivability of α. Two options are considered: migration by occupying another node's position or introducing an additional backup node. In this case, α is adjacent to two parent nodes, δ and β. Both δ and β have already been protected, meaning that, under normal operational conditions without failures, if either of them relocates, the network can continue operating without disruption. This allows α to either migrate to δ's position or β's position.

In the migration option, the first approach α migrates to its parent node δ. Since δ's position is now occupied, δ subsequently relocates to its designated backup node, as illustrated in Fig. 66. In the second approach, α migrates to its parent node β, triggering a chain reaction of migrations similar to previous cases, as depicted in Fig. 71. A comparison of backup resource increments shows that the first approach does not require any additional backup resource increments, whereas the second approach incurs an additional 1 unit of backup resources. As a result, the first approach is selected, as illustrated in Fig. 66.

The backup node option, which involves introducing a backup node, is not further considered in this discussion, as it inherently increases the backup resource increment.

doi:10.6342/NTU202501591

With all virtual nodes protected, the final backup resource increment is determined. The computation backup resource increment amounts to 40, while the bandwidth backup resource increment totals 30, resulting in a total backup resource increment of 70.

Branch 2

In Branch 2, after applying the algorithm process, the transformation of parent and child nodes results in the same structure as in Branch 1. As illustrated in Fig. 61, previously protected child nodes(δ,β,ζ) become parent nodes, while their adjacent, yet unprotected virtual nodes are designated as new child nodes (ε,η,α). Additionally, the order of protecting these child nodes remains the same due to their respective bandwidth demands(ε first, followed by η, and then α).

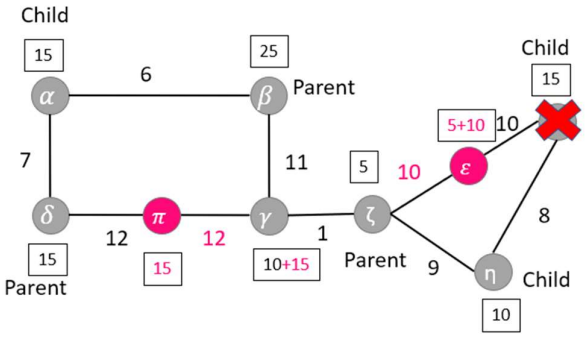


Fig.68 Both options yield the same result for ε's protection

For the protection of ε, it has two possible protection options. It can either migrate to the backup node already placed on the link between ε and the parent node ζ, as shown in Fig. 68, or a new backup node can be introduced on the link with the highest

doi:10.6342/NTU202501591

bandwidth among ε's adjacent links. However, since the highest-bandwidth link is precisely the one connecting ε to ζ, and a backup node already exists there, both options lead to the same result, as illustrated in Fig. 68.

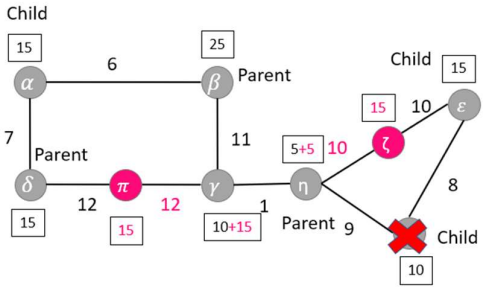


Fig.69 First option for protecting η: occupying ζ 's vacant node

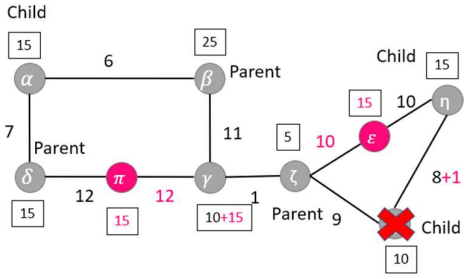


Fig.70 First option for protecting η: occupying ε's vacant node

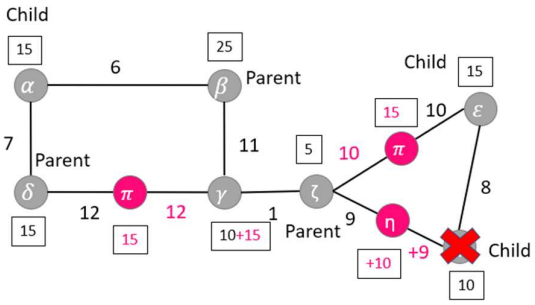


Fig.71 Second option for protecting η

After protecting ε, the next step is to ensure the survivability of η. Two options are evaluated: migration through node occupation and introducing a new backup node. In this case, η is adjacent to its parent node ζ and its sibling node ε. A key observation is that both ζ and ε have already been protected, meaning that, under normal operational conditions, either node can relocate without disrupting service. This allows η to migrate either to its parent node ζ or to the previously protected sibling node ε .

doi:10.6342/NTU202501591

In the migration option, the first approach involves η relocating to its parent node ζ, following the same migration chain as in previous cases, as illustrated in Fig. 69. The second approach involves η occupying the position of ε, which also triggers a sequence of node migrations, as shown in Fig. 70. Comparing backup resource increments, the first approach requires an increment of 5 unit, whereas the second approach incurs an increment of 1 unit. As a result, the second approach is selected, as illustrated in Fig. 70, ensuring efficient resource utilization.

The backup node option introduces a new backup node on the virtual link between η and ζ to provide protection for η, as shown in Fig. 71. The backup resource increments required for these options are compared, with the migration option's second approach requiring 1 unit, while the backup node option requires 19 units. Since the migration option's second approach results in a lower backup resource increment, it is selected, as illustrated in Fig. 70.

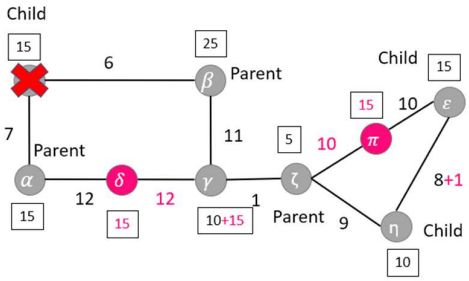


Fig.72 First option for protecting α: occupying δ 's vacant node

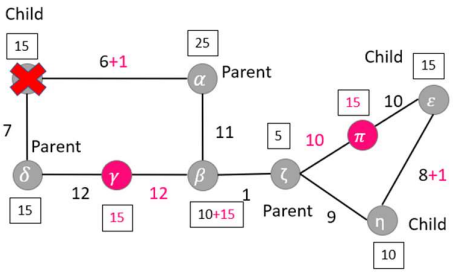


Fig.73 First option for protecting α: occupying β's vacant node

After protecting η, the next step is to ensure the survivability of α. Two options are considered: migration by occupying another node's position or introducing an additional backup node. In this case, α is adjacent to two parent nodes, δ and β. Both δ

doi:10.6342/NTU202501591

and β have already been protected, meaning that, under normal operational conditions without failures, if either of them relocates, the network can continue operating without disruption. This allows α to either migrate to δ's position or β's position.

In the migration option, the first approach α migrates to its parent node δ. Since δ's position is now occupied, δ subsequently relocates to its designated backup node, as illustrated in Fig. 72. In the second approach, α migrates to its parent node β, triggering a chain reaction of migrations similar to previous cases, as depicted in Fig. 73. A comparison of backup resource increments shows that the first approach does not require any additional backup resource increments, whereas the second approach incurs an additional 1 unit of backup resources. As a result, the first approach is selected, as illustrated in Fig. 72.

The backup node option, which involves introducing a backup node, is not further considered in this discussion, as it inherently increases the backup resource increment.

With all virtual nodes protected, the final backup resource increment is determined. The computation backup resource increment amounts to 45, while the bandwidth backup resource increment totals 23, resulting in a total backup resource increment of 68.

While Branch 1 requires a higher total backup resource increment than Branch 2, it also contains one fewer virtual link, specifically the ζ-backup virtual link in Branch 2, which requires 10 units of bandwidth.

doi:10.6342/NTU202501591

A notable aspect is the difference in backup bandwidth increments, where Branch 1 requires 30 units, whereas Branch 2 requires 23 units. This indicates that although Branch 1 incurs an additional 7 units of backup bandwidth increment, the extra virtual link ζ- backup in Branch 2 introduces potential uncertainty in the mapping phase. If this link requires a multi-hop path when embedded onto the SN, the savings in backup resources at the VN level could be offset by increased mapping costs at the SN level.

Since it is impossible to determine the final mapping cost solely based on the backup resource increments of the augmented VN, both Branch 1 and Branch 2 are retained for further evaluation. The final selection between these two configurations will be determined in the mapping phase, where their actual embedding costs can be directly compared.

doi:10.6342/NTU202501591

4.2.3 Pseudo Code for Nest-Base Algorithm

Algorithm 1: Augmented VN Generation (Nest-Based Algorithm)

Input: Original VN

Output: Best_Candidate_Set

```
// Initialize variables (Lines 1--8)
```

1 Migration_Set ← {};
2 Protection_Set ← {};
3 Parent_List ← {};
4 Child_List ← {};
5 Candidate_Set ← {};
6 Best_Candidate_Set ← {};
7 Branch_SET ← new Queue();
8 increment ← 0;

```
// Augment the VN by adding a backup node to protect the first virtual node and
   record the resource increment (Lines 9--13)
```

9 first_node ← SELECT_NODE_WITH_HIGHEST_DEGREE_THEN_MAX_BANDWIDTH(Original VN);
10 virtual_link ← VIRTUAL_LINK_WITH_MAX_BANDWIDTH(first_node.adjacent_links);
11 (*augmented_VN*, *backup_node*) ← PLACE_BACKUP_NODE(Original VN, virtual_link);
12 (*augmented_VN*, *entry*) ← RECOVER_BACKUP_NODE_OPTION(Original VN, augmented_VN, first_node, backup_node);
13 increment ← CALCULATE_INCRMENT(augmented_VN − Original VN);

```
// Record the protection configuration for the first virtual node and mark it as
   protected. Then, designate it as the parent node and assign its adjacent primary
   nodes as children, and sort the protection order for these child nodes (Lines
   14--17)
```

14 Migration_Set ← { *entry* };
15 Protection_Set ← { *first_node* };
16 Parent_List ← { *first_node* };
17 { *Parent_List*, *Child_List* } ← UPDATE_AND_SORT_CHILDLIST(augmented_VN, Protection_Set, Parent_List, Child_List);
18 Branch_SET ← ENQUEUE({ increment, Original VN, augmented_VN, Migration_Set, Protection_Set, Parent_List, Child_List });

```
// Protect all remaining virtual nodes in the augmented VN by applying migration and
   backup strategies, which may result in multiple augmented VNs (Lines 19--38)
```

19 **while** *Branch_SET* ≠ ∅ **do**

```
    // Process branch of the augmented VN
```

20 { increment, Original VN, augmented_VN, Migration_Set, Protection_Set, Parent_List, Child_List } ← DEQUEUE(Branch_SET);

```
    // Process all designated child nodes by evaluating both migration and backup
       options to protect them and update resource increments accordingly (Lines
       21--36)
```


doi:10.6342/NTU202501591

```
21  while Child_List ≠ ∅ do
22      for current_node in (Child_List \ Protection_Set) do
            // Option1: Use migration option to protect child node and record
               resource increment (Lines 23--24)
23          (augmented_VN_migration, entry_migration) ←
            RECOVER_MIGRATIONE_OPTION(Original VN, augmented_VN, current_node,
            Migration_Set, Protection_Set, Parent_List);
24          incrment_migration ← CALCULATE_INCRMENT(augmented_VN_migration −
            augmented_VN);
            // Option2: Add a backup node to protect child node and record resource
               increment (Lines 25--28)
25          virtual_link ← VIRTUAL_LINK_WITH_MAX_BANDWIDTH(
            current_node.adjacent_links);
26          (augmented_VN_backup, backup_node) ← PLACE_BACKUP_NODE(augmented_VN,
            virtual_link);
27          (augmented_VN_backup, entry_backup) ←
            RECOVER_BACKUP_NODE_OPTION(Original VN, augmented_VN_backup,
            current_node, backup_node);
28          incrment_backup ← CALCULATE_INCRMENT(augmented_VN_backup −
            augmented_VN);
29          Protection_Set ← Protection_Set ∪ { current_node };
            // If the resource increment of backup-node option is lower, branch out
               for further processing (Lines 30--33)
30          if incrment_backup < incrment_migration then
31              Backup_Set ← Migration_Set ∪ { entry_backup };
32              incrment_branch ← increment + incrment_backup;
33              Branch_SET ← ENQUEUE({ incrment_branch, Original VN,
                augmented_VN_backup, Backup_Set, Protection_Set, Parent_List, Child_List });
            // Record the migration option augmented VN and update resource increment
               (Lines 34--36)
34          augmented_VN ← augmented_VN_migration;
35          Migration_Set ← Migration_Set ∪ { entry_migration };
36          increment ← increment + incrment_migration;
        // Promote children to parents, define adjacent primary nodes as children,
           then sort
37      { Parent_List, Child_List } ← UPDATE_AND_SORT_CHILDLIST(augmented_VN,
        Protection_Set, Parent_List, Child_List);
    // Record the final augmented VN and resource increment for each branch
38  Candidate_Set ← Candidate_Set ∪ { { increment, augmented_VN } };
// Among augmented VNs with the same quantity of backup node, pick the smallest
   increment as the best candidate and record them
39 Best_Candidate_Set ← SELECT_BEST_CANDIDATE_PER_BACKUP_COUNT(Candidate_Set);
40 return Best_Candidate_Set;
```

Algorithm 1. Pseudo code of Nest-Base algorithm

doi:10.6342/NTU202501591

The pseudo code begins by initializing essential sets and lists (Lines 1-8), including the Migration Set, which records the designated migration target for each virtual node in case the node is affected or its current Nest becomes occupied by another node; the Protection Set, which tracks the list of virtual nodes that have already been protected; and the Parent List and Child List, which specify the parent and child relationships among virtual nodes to guide the order of protection and mapping.

After initialization, the algorithm selects the first virtual node to be protected by first identifying the node with the highest degree. If multiple nodes have the same degree, the node with the largest total surrounding bandwidth demand is selected (Line 9).

The backup node for this first virtual node is then placed along the virtual link with the highest bandwidth demand among its adjacent links (Lines 10-11).

Once the backup node is placed, the corresponding protection configuration is recorded (Line 12), and the resource increment caused by this initial protection is computed (Line 13).

The migration information and protection status of the first protected virtual node are recorded (Lines 14-15), after which this node is designated as the parent node, and its adjacent primary virtual nodes (unprotected virtual neighbors) are identified as child virtual nodes and sorted based on the bandwidth demands of the virtual links connecting them to the parent node (Lines 16-17).

(Lines 19-38) handle the protection process for all augmented VN branches. Each time, a single branch is dequeued for processing (Line 20), and the child virtual nodes

doi:10.6342/NTU202501591

listed in the corresponding Child_List are sequentially processed (Lines 21-36).

Once all child nodes in Child_List are processed, these child nodes are promoted to parent nodes, and their unprotected adjacent primary virtual nodes are identified as the new child nodes. The new child nodes are then sorted to determine the next protection order before the process repeats (Line 37).

At (Line 22), each child virtual node is processed one at a time. For each child node, two protection options are available: Option 1 (Line 23) protects the node through migration, allowing it to occupy an existing Nest; Option 2 (Lines 25-27) protects the node by introducing a new backup node along its adjacent virtual link with the highest bandwidth demand. If Option 2 results in a smaller resource increment compared to Option 1, the algorithm records the current state, along with the augmented VN generated using Option 2, and enqueues this newly formed branch for further processing as an alternative augmented VN branch (Lines 30-33). It is important to note that the decision to adopt Option 2 only determines whether an alternative branch is created, while the current branch always adopts Option 1, and the resulting augmented VN and its corresponding resource increment are recorded (Lines 34-36).

Once all virtual nodes in the current augmented VN branch have been fully protected, the branch is added to the candidate set of augmented VNs (Line 38).

After all branches have been processed, the algorithm first categorizes the candidate branches based on the number of backup nodes they contain. Among the candidates with the same number of backup nodes, the branch with the smallest resource increment is selected as the best augmented VN for that particular backup node count.

doi:10.6342/NTU202501591

As a result, the Best_Candidate_Set may contain multiple augmented VNs, each corresponding to a different number of backup nodes. This complete set of best candidates is then returned as the output (Lines 39-40).

4.3 Design for Mapping Algorithm

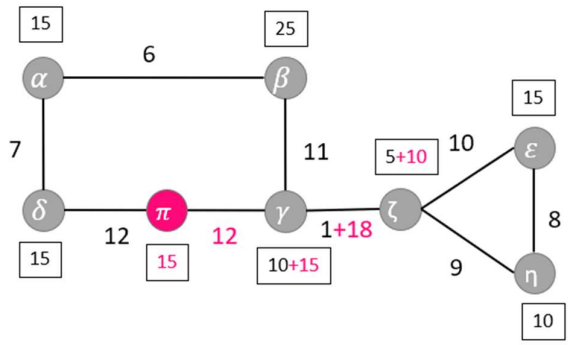
4.3.1 Mapping Process for Augmented VN

To efficiently minimize embedding costs on large-scale SN during the mapping phase, the backup resources of the augmented VN must first be optimized at the VN level. This optimization relies on the assumption that "an augmented VN with a smaller resource increment will lead to a lower embedding cost." For this assumption to hold, the mapping algorithm must exhibit linearity.

As discussed in Section 2.3.1, the mapping algorithms proposed in [16] and [17] lack the property of linearity, which prevents them from maintaining a linear relationship between the augmented VN's resource increment and the final embedding cost. This lack of linearity stems from two key issues: (1) these algorithms consider the sharing of backup resources directly within the SN during the mapping phase, and (2) they do not ensure that all virtual links are mapped in a way that prioritizes 1-hop paths whenever possible.

To address these issues and ensure the linearity required by our design objective, the proposed mapping algorithm is explicitly designed to possess these two essential principles: (1) backup resource sharing is excluded from the mapping phase, and (2) the link mapping process aims to achieve 1-hop paths for virtual links as much as possible.

doi:10.6342/NTU202501591



VN node	Surrounding bandwidth demand
α	6+7=13
β	6+11=17
γ	**11+12+19=42**
δ	7+12=19
ε	8+10=18
ζ	9+10+19=38
η	8+9=17
π	12+12=24

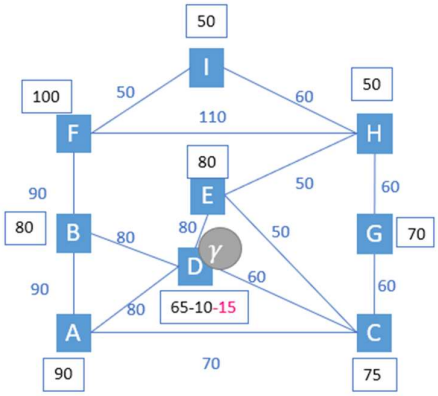
Fig.74 Nest-Base: Augmented VN and virtual node metric values

To illustrate the operational process of the mapping algorithm, we consider the augmented VN obtained from Branch 1 in Section 4.2, as depicted in Fig. 74. The mapping process follows a sequential approach, where virtual nodes are mapped one at a time, and virtual links are embedded immediately after both of their endpoint virtual nodes are mapped. The shortest path algorithm is used to determine the mapping of virtual links.

The mapping process begins by selecting the first virtual node to be embedded. To determine the starting point, each virtual node's surrounding bandwidth demand is calculated as the sum of the bandwidth demands of all adjacent virtual links, as illustrated in Fig. 74. Among all virtual nodes, γ has the highest total surrounding bandwidth demand, making it the most critical virtual node for embedding. Since the total embedding cost is largely determined by path mapping costs, and the placement of virtual nodes directly influences the mapping path lengths of their virtual links, prioritizing γ ensures that the virtual node with the highest surrounding bandwidth demand is strategically placed in a best position. This facilitates efficient path

doi:10.6342/NTU202501591

mapping for subsequent nodes and minimizes overall embedding costs. Thus, γ is chosen as the first node to be mapped.



Substrate node	metric
A	3*(90+80+70)=720
B	3*(90+80+90)=780
C	4*(70+60+50+60)=960
D	4*(80+80+80+60)=1200
E	3*(80+50+50)=540
F	3*(50+90+110)=750
G	2*(60+60)=240
H	4*(60+110+50+60)=1120
I	2*(50+60)=220

Fig.75 Mapping the first virtual node to the substrate node with the highest metric value

Once the first virtual node is determined, it is embedded onto the substrate node with the highest metric in the SN, as shown in Fig. 75. It is important to note that the SN used in this example is the same as the one shown in Fig. 13; however, the calculation of the substrate node metric differs. The substrate node metric is computed by multiplying the degree of the node by the sum of its adjacent available bandwidths. Following this approach, the virtual node γ is mapped onto the substrate node D, which has the highest metric value, as its high degree indicates a greater number of directly connected substrate nodes, and its high available bandwidth sum ensures that sufficient resources are available to meet future path-mapping requirements.

The rationale behind this strategy is that embedding the virtual node with the highest surrounding bandwidth demand onto the substrate node with the highest metric value ensures that the subsequent virtual nodes surrounding γ can be more easily mapped

doi:10.6342/NTU202501591

onto nearby substrate nodes. This increases the likelihood that their virtual links will require only 1-hop paths in the link mapping process, reducing overall path mapping costs.

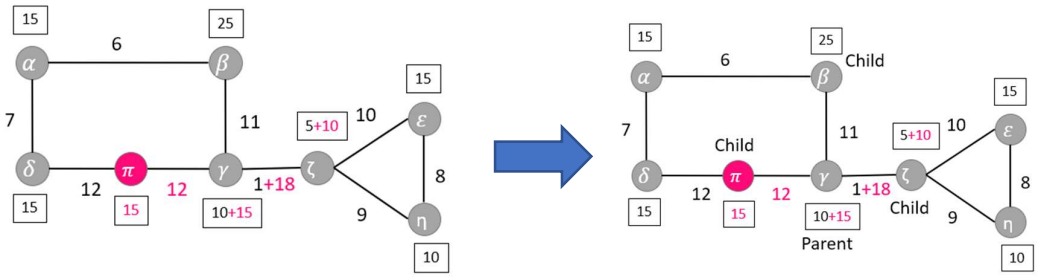


Fig.76 Defining γ's adjacent nodes as child nodes during mapping phase

Once γ has been mapped, all its adjacent virtual nodes are designated as child nodes, while γ itself is classified as a parent node, as illustrated in Fig.76.

The reason for defining the newly mapped virtual node as a parent node and its adjacent virtual nodes as child nodes is to ensure that in the next step, when mapping these child nodes (π,β, ζ), they can be mapped to the substrate nodes that are closest to the substrate node where the parent node (γ) has been mapped, which is D. By mapping child nodes near the parent node in the SN, the subsequent virtual link mappings are more likely to achieve 1-hop path length. By maintaining this 1-hop correspondence, the method preserves the linearity property of the mapping process, while also achieving the minimum path mapping cost.

The mapping order of child nodes is determined based on the bandwidth demand of the virtual link between each child node and the parent node (γ), sorted in descending

doi:10.6342/NTU202501591

order. By following this principle, ζ is mapped first, followed by π, and finally β. Prioritizing the mapping of child nodes with higher bandwidth demands ensures that these virtual nodes are placed at the most suitable substrate nodes, allowing the corresponding high-bandwidth virtual links to be mapped efficiently with 1-hop paths. This helps minimize the overall path mapping cost.

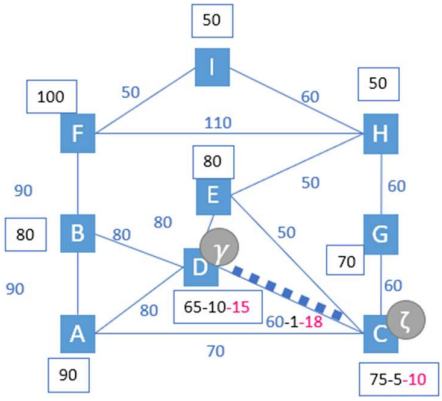


Fig.77 Mapping ζ and its virtual link to γ

When mapping ζ, each available substrate node is tested one by one. After ζ is mapped to a substrate node, the corresponding virtual link between ζ and γ is immediately mapped to ensure that the embedding cost can be accurately calculated. After evaluating all possible substrate nodes, it is found that mapping ζ to A, B, C, or E results in the same minimum embedding cost. This is because these substrate nodes allow the virtual link between ζ and γ to be embedded with a 1-hop path, while also having sufficient available computational resources to accommodate ζ.

Since multiple substrate nodes provide the same minimum embedding cost, the final selection is determined based on the metric values from Fig.75. The metric values for A, B, C, and E are 720, 780, 960, and 540, respectively. Among these candidates, C

doi:10.6342/NTU202501591

has the highest metric value, making it the best choice for mapping ζ. Once ζ is mapped to C, the virtual link between ζ and γ is immediately mapped onto the SN, as illustrated in Fig.77.

During the link mapping process, the primary bandwidth and backup bandwidth are directly allocated, as indicated by the black and pink text in Fig.77, respectively. The mapping does not consider backup bandwidth sharing among backup paths in the SN, ensuring that the total backup bandwidth is explicitly allocated for each virtual link. Ensuring that the mapping algorithm maintains linearity by preventing unpredictable bandwidth sharing effects that could alter the relationship between the backup resource increments in the augmented VN and the resulting embedding cost.

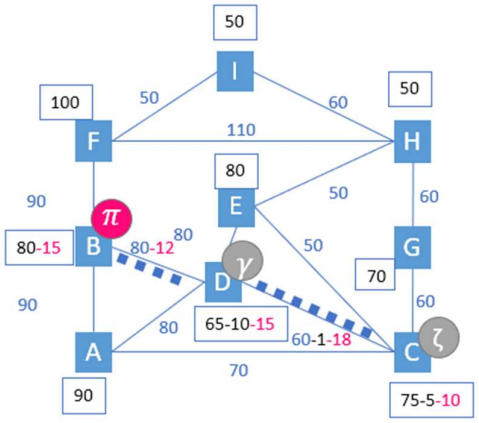


Fig.78 Mapping backup node and its virtual link to γ

Following the same mapping strategy, π is tested on all available substrate nodes. After each mapping attempt, the corresponding virtual link between π and γ is immediately mapped to evaluate the embedding cost. After testing all possible substrate nodes, it is found that mapping π to A, B, or E results in the same minimum embedding cost, as all three substrate nodes allow the virtual link between π and γ to

doi:10.6342/NTU202501591

be embedded with a 1-hop path, and their available computational resources are sufficient to accommodate π.

Since multiple substrate nodes provide the same minimum embedding cost, the final selection is determined based on the metric values from Fig.75. The metric values for A, B, and E are 720, 780, and 540, respectively. Among these candidates, B has the highest metric value, making it the optimal choice for mapping π. Once π is mapped to B, the virtual link between π and γ is immediately mapped onto the SN, as illustrated in Fig.78.

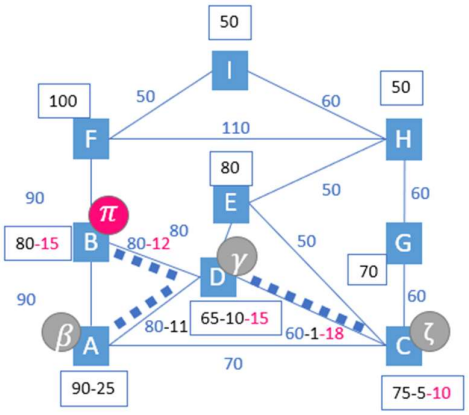


Fig.79 Mapping β and its virtual link to γ

Following the same mapping approach, β is tested on all available substrate nodes, and each time it is mapped, the corresponding virtual link between β and γ is immediately embedded to evaluate the embedding cost. After testing all possible substrate nodes, it is found that mapping β to A or E results in the same minimum embedding cost. Both substrate nodes allow the virtual link between β and γ to be embedded with a 1-hop path, and their available computational resources are sufficient to support β.

doi:10.6342/NTU202501591

Since multiple substrate nodes provide the same minimum embedding cost, the final selection is determined based on the metric values from Fig.75. The metric values for A and E are 720 and 540, respectively. Among these candidates, A has the highest metric value, making it the optimal choice for mapping β. Once β is mapped to A, the virtual link between β and γ is immediately mapped onto the SN, as illustrated in Fig.79.

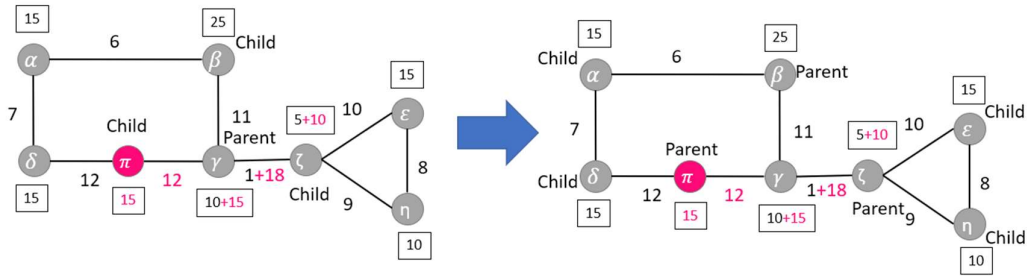


Fig.80 Parent-child node transition during mapping phase

After mapping ζ, π, and β, the next step is to update the parent-child relationships among the virtual nodes. As shown in Fig.80, the newly mapped nodes ζ, π, and β are now designated as parent nodes, while their adjacent virtual nodes that have not yet been mapped, α, δ, ε, and η, are assigned as child nodes, ensuring that they will be mapped next.

Since these child nodes are now ready for mapping, their mapping order must be determined to optimize the overall embedding efficiency. The mapping order of child nodes is based on the bandwidth demand of the virtual link between each child node and its parent node, sorted in descending order. Following this criterion, the child nodes are mapped in the order of δ, ε, η, and α.

doi:10.6342/NTU202501591

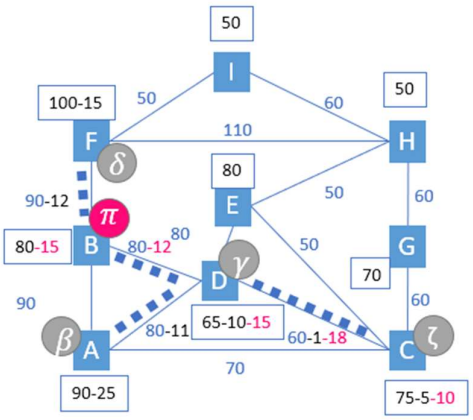


Fig.81 Mapping δ and its virtual link to backup node

The mapping of δ follows the same approach as other child nodes. Each possible substrate node is tested one by one. After mapping δ to a substrate node, the corresponding virtual link between δ and its parent node π is immediately mapped to evaluate the embedding cost. This process is repeated until all possible substrate nodes have been tested.

After testing all potential substrate nodes, the optimal choice is determined by selecting the substrate node that is closest to the substrate node where the parent node π has been mapped (B), while also having sufficient available computational resources. Based on this criterion, F is selected as the substrate node for δ, as it meets both conditions. Once δ is mapped to F, the virtual link between δ and π is immediately mapped onto the SN, as illustrated in Fig.81.

doi:10.6342/NTU202501591

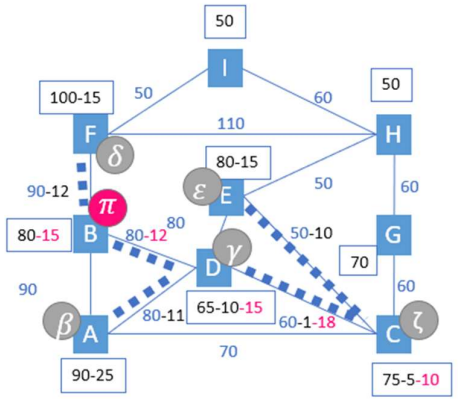


Fig.82 Mapping ε and its virtual link to ζ

After mapping δ, the next step is to map ε following the same approach as the previous child nodes. Each possible substrate node is tested one by one. After mapping ε to a substrate node, the corresponding virtual link between ε and its parent node ζ is immediately mapped to evaluate the embedding cost. This process is repeated until all possible substrate nodes have been tested.

The optimal substrate node for ε is determined by selecting the node that is closest to the substrate node where the parent node ζ has been mapped (C), while also having sufficient available computational resources. After testing all possible substrate nodes, E and G are found to have the same minimum embedding cost. Since multiple candidates provide an optimal mapping cost, the final selection is made based on the metric values from Fig.75. The metric values for E and G are 540 and 240, respectively, making E the optimal choice for mapping ε. Once ε is mapped to E, the virtual link between ε and ζ is immediately mapped onto the SN, as illustrated in Fig.82.

doi:10.6342/NTU202501591

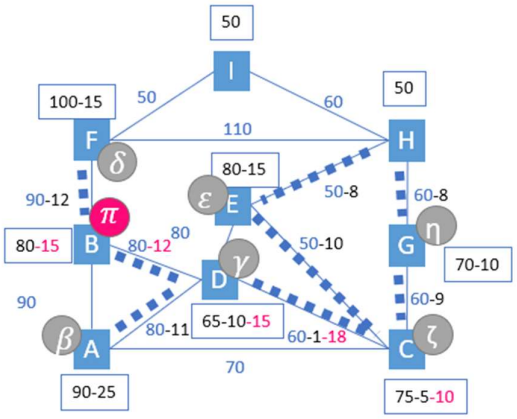


Fig.83 Mapping η and its virtual links to ζ and ε

After mapping ε, the next step is to map η. Since η's adjacent virtual nodes include both its parent node ζ and its child node ε, which have already been mapped, its mapping process differs from previous child node mappings. Unlike the previous cases where only the parent node was considered, η's mapping must take into account both ζ and ε to ensure an optimal placement. As with other mappings, η is tested on all possible substrate nodes one by one. After mapping η to a substrate node, the corresponding virtual links between η and both ζ and ε are immediately mapped to evaluate the total embedding cost. This process is repeated until all possible substrate nodes have been tested.

The optimal substrate node for η is determined by selecting the node that is closest to both the substrate node where its parent node ζ has been mapped (C) and the substrate node where its adjacent child node ε has been mapped (E), while also having sufficient available computational resources. Here, "closest" refers to the substrate node that results in the minimum embedding cost when mapping the virtual links between η and both ζ and ε.

doi:10.6342/NTU202501591

Based on this criterion, G is selected as the substrate node for η since it has the lowest embedding cost, with a path cost of 9 to ζ and 8*2 to ε. Once η is mapped to G, the virtual links between η and both ζ and ε are immediately mapped onto the SN, as illustrated in Fig.83.

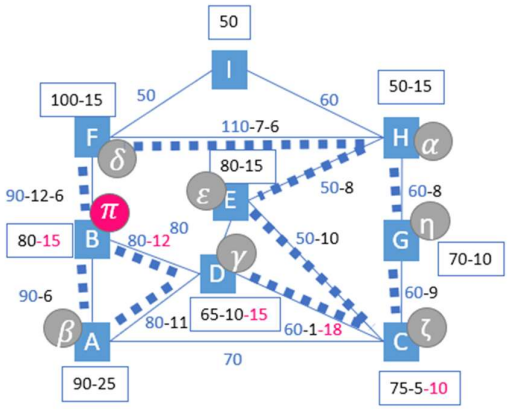


Fig.84 Mapping α and its virtual links to β and δ

After mapping η, the next step is to map α. Unlike previous child node mappings, α's mapping process must consider two previously mapped parent nodes, β and δ, instead of just one. Since β has been mapped to A and δ has been mapped to F, α must be placed at a substrate node that is closest to both A and F while also having sufficient available computational resources.

As with other mappings, α is tested on all possible substrate nodes one by one. After each mapping attempt, the corresponding virtual links between α and both β and δ are immediately mapped to evaluate the total embedding cost. This process is repeated until all possible substrate nodes have been tested. After testing all candidates, H and I are found to have the same minimum embedding cost. The final selection is determined based on the metric values from Fig.75, where H has a metric value of

doi:10.6342/NTU202501591

1120 and I has a metric value of 220. Since H has the highest metric value, it is chosen as the final mapping location for α. Once α is mapped to H, the virtual link between α and both β and δ are immediately mapped onto the SN, as illustrated in Fig.84.

The mapping results indicate that most virtual links were successfully embedded using 1-hop paths, aligning with the intended strategy to minimize total path mapping costs. However, due to the limited availability of SN resources, some virtual nodes had to be mapped to SN locations that resulted in longer mapping paths. Specifically, the virtual link between ε and η requires a 2-hop SN paths, while the virtual link between α and β requires a 3-hop SN paths. A key observation is that these virtual links have some of the lowest bandwidth demands among all virtual links in the augmented VN, with demands of only 8 and 6, respectively. These values rank as the third smallest and the smallest bandwidth demands, meaning that even though these mapping paths are longer, their low bandwidth demands prevent them from significantly impacting overall bandwidth consumption.

While achieving 1-hop paths for all virtual links is the ideal scenario, this mapping algorithm is not only beneficial for optimizing bandwidth allocation but also crucial for maintaining the linearity of the mapping algorithm. However, practical SN resource constraints make it impossible to achieve 1-hop paths for every virtual link. In such cases, rather than allowing high-bandwidth virtual links to be mapped onto longer SN paths, it is more cost-effective to let low-bandwidth virtual links bear this additional path length. This strategic trade-off minimizes the overall bandwidth allocation cost while still preserving the linear properties of the mapping process.

doi:10.6342/NTU202501591

Since low-bandwidth virtual links have a smaller impact on total bandwidth consumption, placing them on longer SN paths when necessary is a more practical and cost-efficient choice to maintain overall mapping efficiency.

Comparison Between Branch 1 and Branch 2

Following the mapping process, the total embedding costs for Branch 1 and Branch 2 are evaluated based on computational resource consumption and bandwidth resource consumption. The results show that Branch 1 incurs a computational resource consumption of 135 and a bandwidth resource consumption of 114, leading to a total embedding cost of 249. On the other hand, Branch 2 requires 140 for computational resources and 121 for bandwidth resources, resulting in a total embedding cost of 261, as shown in Fig.85. Since the mapping process for Branch 2 follows the same methodology as Branch 1, the detailed mapping steps are omitted. Given that Branch 1 achieves a lower total embedding cost compared to Branch 2, it is selected as the final mapping result.

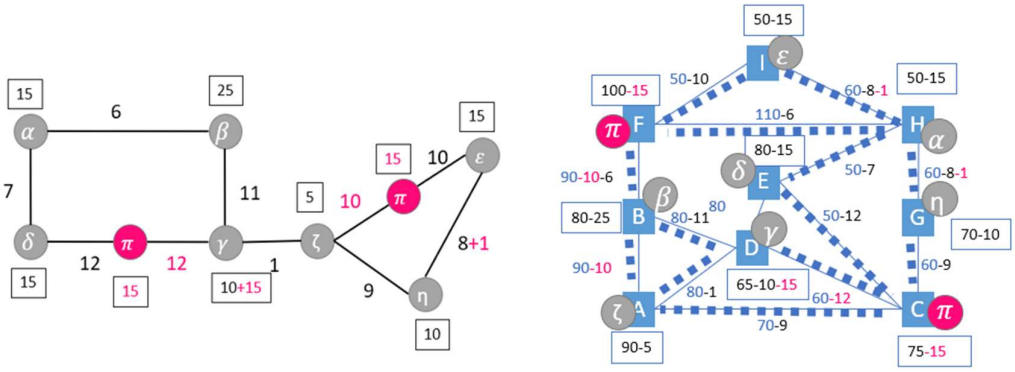


Fig.85 Mapping result of the second branch's augmented VN

doi:10.6342/NTU202501591

The embedding costs of the selected Branch 1 mapping result are compared with those of the methods presented in [16] and [17]. The FIP method from [16] has a total embedding cost of 267, as shown in Fig. 86.

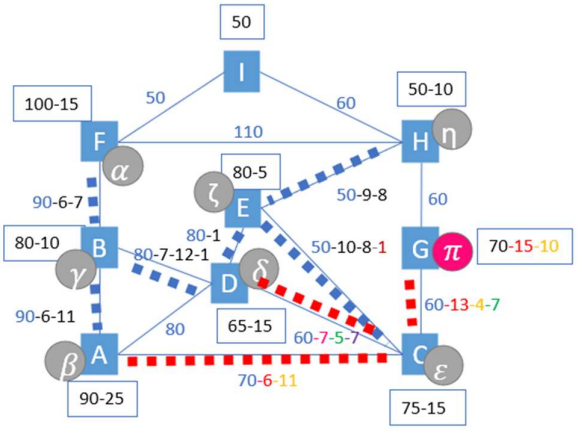
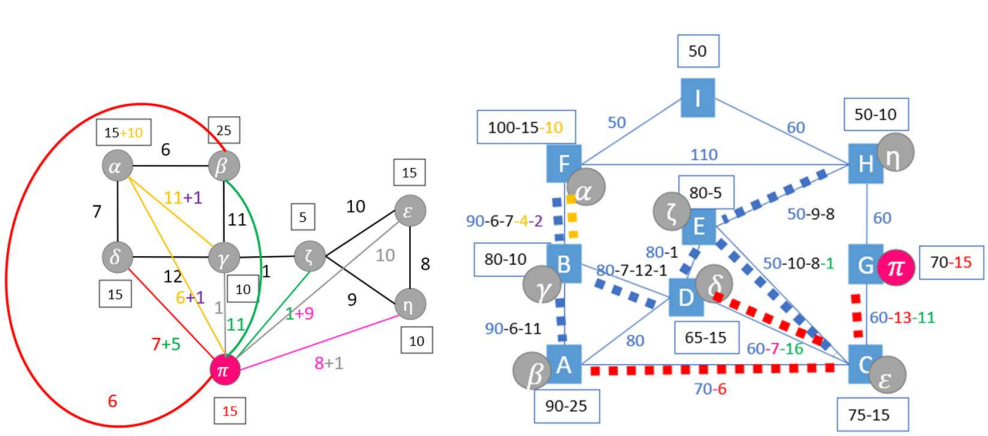


Fig.86 Mapping result of the 7-node VN in Fig. 51 using FIP

While the FDP method from [16] results in an embedding cost of 266, as illustrated in Fig. 87. Additionally, Fig. 87 presents the backup resource increment required for each node failure scenario using different colors in the table.

doi:10.6342/NTU202501591



VN node	Color
α	Red
β	Orange
γ	Green
δ	Purple
ε	Pink
ζ	Gray
η	-

Fig.87 Mapping result of the 7-node VN in Fig. 51 using FDP, with different backup resource increments indicated by colors

In contrast, the augmented VN method from [17] fails to complete the embedding process due to the VN consisting of 10 virtual nodes, which exceeds the 9 substrate nodes available in the SN, as depicted in Fig. 88.

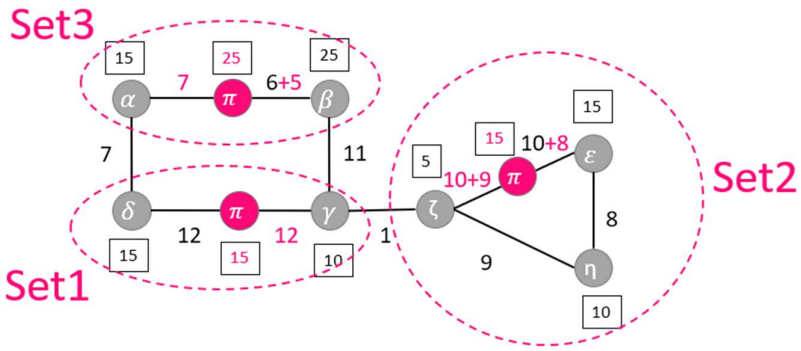


Fig.88 Augmented 7-node VN in Fig. 51 using ProRed

doi:10.6342/NTU202501591

These results demonstrate that the proposed approach, represented by the Branch 1 mapping result, achieves a lower embedding cost than both the FIP and FDP methods in [16]. Furthermore, compared to the augmented VN method from [17], which fails due to an excessive number of virtual nodes, the proposed method successfully completes the embedding process without exceeding the available substrate nodes.

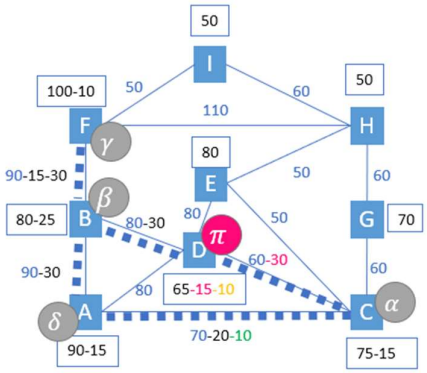


Fig.89 Augmented 4-node VN in Fig. 1 using our proposed algorithm

To further validate the effectiveness of the proposed mapping algorithm, we also apply it to the same 4-node VN topology used in Section 2.2.2.2, where the mapping algorithms in [16] and [17] were evaluated. This allows a fair comparison between the proposed method and existing works under identical VN conditions. The resulting mapping cost for the proposed algorithm is 255, as shown in Fig.89, which is significantly lower than the mapping costs of 330 and 315 achieved by the FIP and FDP algorithms in [16], respectively, and also lower than the mapping cost of 305 achieved by the ProRed algorithm in [17]. These results further demonstrate that the proposed algorithm achieves superior embedding performance.

doi:10.6342/NTU202501591

4.3.2 Pseudo Code for Mapping Algorithm

Algorithm 2: Mapping Augmented VN to SN (Mapping Algorithm)

Input: SN, Best_Candidate_Set
Output: Optimal_Mapping_Solution

```
1  Solution_Set ← {};
   // Compute a metric value for each substrate node in SN
2  Node_Metric_Values ← CALCULATE_NODE_METRIC_VALUES(SN);
   // Mapping each candidate augmented VN solution independently onto SN_Temp, which is
      an identical but independent copy of SN (Line 3--45)
3  for {increment, augmented_VN} ∈ Best_Candidate_Set do
       // Initialize mapping variables for this candidate augmented VN (Lines 4--10)
4      Parent_List ← {};
5      Child_List ← {};
6      Mapped_Substrate_Node_List ← {};
7      Mapped_Virtual_Node_List ← {};
8      Node_Mapping_Solution_Set ← {};
9      Link_Mapping_Solution_Set ← {};
10     total_mapping_cost ← 0;
       // Create a copy of SN for independent mapping of this candidate augmented VN
          (Line 11)
11     SN_Temp ← SN;
       // Map the first virtual node: select the virtual node with the highest
          surrounding bandwidth, then map it to the most appropriate substrate node;
          record the mapping and mapping cost. Reject the candidate if no substrate
          node is available (Lines 12--19)
12     first_virtual_node ←
       VIRTUAL_NODE_WITH_HIGHEST_SURROUNDING_BANDWIDTH(augmented_VN);
13     selected_substrate_node ← SORT_AND_SELECT_SUBSTRATE_NODE(first_virtual_node,
       Node_Metric_Values, SN_Temp);
14     if selected_substrate_node = Null then
15         continue;
16     node_mapping_cost ← MAP_VIRTUAL_NODE(first_virtual_node, selected_substrate_node,
       SN_Temp);
17     Mapped_Substrate_Node_List ← Mapped_Substrate_Node_List ∪ {selected_substrate_node};
18     Node_Mapping_Solution_Set ← Node_Mapping_Solution_Set ∪ {{first_virtual_node,
       selected_substrate_node}};
19     total_mapping_cost ← node_mapping_cost;
```


doi:10.6342/NTU202501591

```
    // Set the first virtual node as the parent and define its adjacent virtual
       nodes as children; then sort the child mapping order (Lines 20--21)
20  Parent_List ← {first_virtual_node};
21  {Parent_List, Child_List} ← UPDATE_AND_SORT_CHILDLIST(augmented_VN,
    Mapped_Virtual_Node_List, Parent_List, Child_List);
    // Use a boolean flag to indicate this candidate augmented VN mapping failure
22  mapping_failed ← false;
    // Map the remaining virtual nodes and all virtual links onto SN (Line 23--42)
23  while Child_List ≠ ∅ do
24      for current_virtual_node in Child_List do
25          Candidate_Node_And_Link_Mapping_Set ← {};
            // Test mapping of the current virtual node to every available substrate
               node and record the link mapping cost and mapping path set (Lines
               26--28)
26          for current_substrate_node in (SN_Temp.nodes \ Mapped_Substrate_Node_List) do
27              link_mapping_cost, Mapping_Path_Set ←
                TEST_NODE_AND_LINK_MAPPING(current_virtual_node, current_substrate_node,
                Node_Mapping_Solution_Set, SN_Temp, augmented_VN);
28              Candidate_Node_And_Link_Mapping_Set ← Candidate_Node_And_Link_Mapping_Set
                ∪ {{link_mapping_cost, current_substrate_node, Mapping_Path_Set}};

            // Select the substrate node with the lowest link mapping cost; if
               multiple substrate nodes have the same cost, choose the one with the
               highest node metric value.  Then, map the current virtual node and its
               corresponding virtual links.  Reject the candidate if node or link
               mapping fails (Lines 29--39)
29          if Candidate_Node_And_Link_Mapping_Set = ∅ then
                // if node or link mapping fails, set the flag to true
30              mapping_failed ← true;
31              break;
32          else
33              selected_substrate_node, Mapping_Path_Set ←
                THE_BEST_CANDIDATE_MAPPING_SET(Candidate_Node_And_Link_Mapping_Set,
                Node_Metric_Values);
34              node_mapping_cost ← MAP_VIRTUAL_NODE(current_virtual_node,
                selected_substrate_node, SN_Temp);
35              link_mapping_cost ← MAP_VIRTUAL_LINK(Mapping_Path_Set, SN_Temp);
36              Mapped_Virtual_Node_List ← Mapped_Virtual_Node_List ∪ {current_virtual_node};
37              Node_Mapping_Solution_Set ← Node_Mapping_Solution_Set ∪
                {{current_virtual_node, selected_substrate_node}};
38              Link_Mapping_Solution_Set ← Link_Mapping_Solution_Set ∪ {Mapping_Path_Set};
39              total_mapping_cost ← total_mapping_cost + node_mapping_cost +
                link_mapping_cost;
```


doi:10.6342/NTU202501591

```
        // If node or link mapping fails, terminate the mapping of this augmented VN
40      if mapping_failed = true then
41          break;
        // Update parent-child relationships for the current candidate augmented VN
            (Line 42)
42      {Parent_List, Child_List} ←
        UPDATE_AND_SORT_CHILDLIST(Mapped_Virtual_Node_List, Parent_List, Child_List,
        augmented_VN);
    // If node or link mapping fails, reject this candidate augmented VN
43  if mapping_failed = true then
44      continue;
    // Record the mapping solution and total mapping cost for this candidate
       augmented VN (Line 45)
45  Solution_Set ← Solution_Set ∪ {{total_mapping_cost, Node_Mapping_Solution_Set,
    Link_Mapping_Solution_Set}};
// If no valid mapping solution exists among all candidate augmented VNs, reject
   this VN Request; otherwise, return the optimal mapping solution with the minimum
   mapping cost (Line 46--49)
46 if Solution_Set = ∅ then
47     return False;
48 Optimal_Mapping_Solution ← SELECT_OPTIMAL_MAPPING_SOLUTION(Solution_Set);
49 return Optimal_Mapping_Solution;
```

Algorithm 2. Pseudo code of mapping algorithm

A set is initialized to store the mapping costs and corresponding mapping solutions for all input candidate augmented VNs (Line 1). The metric value calculated for each substrate node determines the node mapping target for the first virtual node and helps select among equally optimal substrate nodes during subsequent node mappings (Line 2).

Each candidate augmented VN is mapped onto an independent copy of SN, and both its mapping cost and corresponding mapping solution are stored in the Solution_Set (Lines 3-45).

Mapping variables are initialized to support the mapping process (Lines 4-10), including the parent and child node lists, the sets of mapped substrate and virtual

doi:10.6342/NTU202501591

nodes, the node-to-substrate mapping records, the link mapping paths, and the accumulated total mapping cost. The SN is then duplicated to create an independent copy, SN_Temp, which is identical to the original SN but mapped separately (Line 11).

The first virtual node to be mapped is selected based on the total bandwidth demand of its adjacent virtual links, and it is mapped to the substrate node with the highest metric value in Node_Metric_Values that also satisfies the node computing resource demand. The mapped substrate nodes, and node mapping cost are recorded accordingly. If no suitable substrate node is available, this candidate augmented VN is discarded (Lines 12-19).

The first mapped virtual node is designated as the parent node, and its adjacent virtual nodes are identified as child nodes. These child nodes are then sorted to determine the order in which they will be mapped during the mapping phase (Lines 20-21). A flag is also initialized to indicate whether node or link mapping failures occur during the mapping process (Line 22).

The remaining virtual nodes and their corresponding virtual links in the candidate augmented VN are mapped onto the SN (Lines 23-42). This process maps one virtual node at a time through an iterative loop (Line 24). For each virtual node, all substrate nodes that satisfy the node computing resource demands are tested, and a trial mapping is performed for both the node itself and the virtual links connecting it to previously mapped nodes in the Node_Mapping_Solution_Set (Lines 26-28). The link mapping cost for each trial is recorded. In (Lines 29-31), if no valid substrate node can be found, indicating that either node or link mapping has failed, the algorithm sets

doi:10.6342/NTU202501591

mapping_failed = True and immediately breaks out of the loop. Otherwise, the substrate node with the smallest link mapping cost is selected. If multiple candidates have the same cost, the one with the highest metric value in Node_Metric_Values is chosen. The mapping information for both the virtual node and the corresponding substrate node is updated, and the total mapping cost is accumulated accordingly (Lines 32-39).

If a node or link mapping failure occurs, the algorithm immediately terminates the candidate augmented VN mapping process for the remaining virtual nodes and links (Lines 40-41). The failed candidate augmented VN is then discarded without being added to the solution set (Lines 43-44). If the mapping succeeds, the algorithm updates the parent-child relationships by promoting the mapped child node to a parent node and directly updating the child list to include the unmapped neighboring virtual nodes of the new parent node. The updated child list is then re-sorted to determine the subsequent mapping order (Line 42).

After mapping all candidates, the algorithm selects the solution with the lowest mapping cost as the final output. If no candidate augmented VN can be successfully mapped, the original VNR is rejected (Lines 46-49).

doi:10.6342/NTU202501591

CHAPTER 5
PERFORMANCE EVALUATION

This chapter evaluates the effectiveness of the proposed augmentation and mapping algorithms through numerical analysis using exhaustive search and simulation, with the numerical analysis conducted under small-scale conditions and the simulation modeling realistic network scenarios. Section 5.1 conducts numerical analysis using exhaustive search to verify the linearity of the proposed mapping algorithm and to compare the lowest mapping cost obtained under the MUPL augmentation strategy with those achieved by other augmentation strategies, across different SN densities and various VN topologies. The evaluation is conducted using small-scale VNs with four virtual nodes, covering all possible VN topologies, to ensure a comprehensive assessment of the augmentation and mapping performance. Section 5.2 evaluates the practical performance of the proposed methods in realistic network scenarios, considering different SN sizes and workload conditions, measuring mapping (embedding) cost under moderate load and acceptance rate under high load conditions.

5.1 Numerical Analysis

This section conducts exhaustive numerical analysis under different SN densities to validate the linearity property of the proposed mapping algorithm and to evaluate the effectiveness of the MUPL augmentation strategy in identifying the augmented VN with the minimum increment and the lowest mapping cost. The evaluation also includes comparisons with existing augmentation strategies, SUDN ([16]) and MRPL ([17]).

doi:10.6342/NTU202501591

For each numerical analysis, a fixed VN topology is selected from the six possible topologies with four virtual nodes. Under each augmentation strategy, all possible augmented VNs are exhaustively generated (except in the MUPL strategy, where only single-backup-node cases are considered due to combinatorial explosion). Each generated augmented VN is then mapped onto an independent but identical SN using the mapping algorithm adopted in the corresponding reference—FIP or FDP for SUDN ([16]), ProRed for MRPL ([17]), and the proposed linear mapping algorithm for MUPL. Both the augmentation increment and the resulting mapping cost are recorded for every configuration.

In addition to the overall evaluation, the numerical analysis also assesses whether the proposed Nest-Base algorithm can effectively identify the minimal-increment augmented VN under the MUPL strategy. Furthermore, the mapping cost of the augmented VN selected by the Nest-Base algorithm is compared against the mapping costs of the augmented VNs selected by the heuristic methods used in [16] and [17], to evaluate the overall efficiency and quality of the proposed solution.

Due to the exponential growth of the augmentation solution space in MUPL, performing a complete exhaustive search across all possible augmented VNs is infeasible. For instance, as shown in Section 4.1, even a small-scale VN with four virtual nodes and four virtual links can result in 497,871,360,000 possible augmented VN configurations under MUPL, making full enumeration impractical. To address this, the exhaustive numerical analysis is conducted using a single backup node for augmentation, which remains a reasonable approach for small-scale VNs. This is based on the assumption that, in small-scale VNs(four virtual nodes), the optimal augmented VN is likely to involve only one backup node, as adding multiple backup

doi:10.6342/NTU202501591

nodes would require additional computational resources, backup links, and bandwidth, which may not contribute to achieving the minimum increment augmented VN.

The experimental setup is detailed in Section 5.1.1, which describes the VN and SN configurations, including the selection of different SN density levels. The results are categorized based on three SN density levels: sparse, semi-dense, and dense, which are analyzed in Sections 5.1.2, 5.1.3, and 5.1.4, respectively. Finally, Section 5.1.5 summarizes the key observations from these results, providing a consolidated analysis of the effectiveness of the proposed method across different SN densities.

5.1.1 Experimental Setup

This section details the experimental setup used in the exhaustive numerical analysis. Section 5.1.1.1 describes the VN settings, including the six predefined VN topologies that ensure a comprehensive evaluation across different network structures. Section 5.1.1.2 outlines the SN configurations, categorized into three different density levels—sparse, semi-dense, and dense—each representing different levels of connectivity and resource availability. Section 5.1.1.3 introduces the performance metrics used to evaluate the augmentation and mapping methods, focusing on increment difference to assess the efficiency of the augmentation strategy, mapping (embedding) cost difference to evaluate the overall performance of the designed algorithm, and correlation coefficient to analyze the linearity of the mapping algorithm.

doi:10.6342/NTU202501591

5.1.1.1 Virtual Network Settings

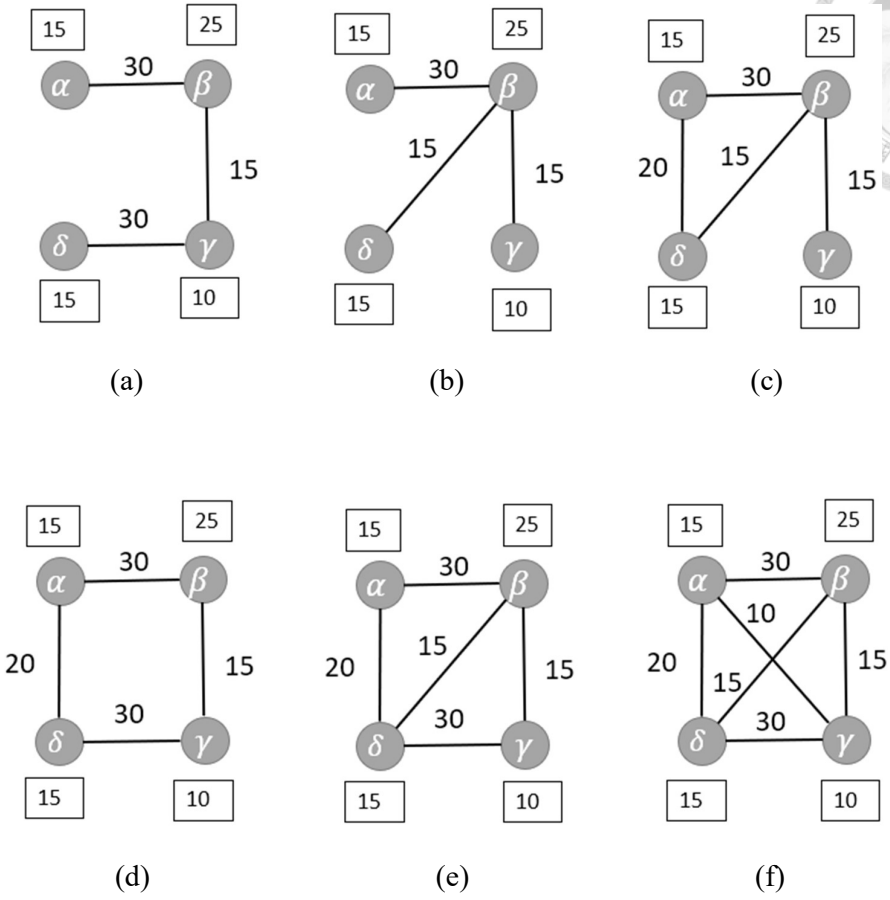


Fig. 90 VN topology: (a) Line. (b) Star. (c) Star+1. (d) Ring. (e) Fully Connected-1. (f) Fully Connected.

Each VN used in the exhaustive numerical analysis consists of four virtual nodes and follows one of six predefined topologies, ensuring a comprehensive evaluation of different VN topologies. The six topologies used are as follows. The Line topology (Fig. 90 (a)) consists of four nodes connected in a straight path, where each node is linked to at most two other nodes. The Star topology (Fig. 90 (b)) has a central node connected to all other nodes, forming a hub-and-spoke pattern. The Star with an Extra Link topology (Fig. 90 (c)) extends a basic star formation by adding an extra link. The Ring topology (Fig. 90 (d)) forms a closed-loop where each node connects to exactly

doi:10.6342/NTU202501591

two neighboring nodes. The Fully Connected Minus One Link topology (Fig. 90 (e)) resembles a complete mesh but with one missing link, creating a near-complete interconnection. Lastly, the Fully Connected topology (Fig. 90 (f)) represents a scenario where every node is directly connected to all other nodes.

These six topologies represent all possible VN topologies with four nodes, ensuring that the exhaustive numerical analysis considers every potential VN configuration. By using all possible VN topologies, we validate whether our proposed methods remain effective across different VN shapes. The bandwidth and computational resource demands for each topology are explicitly defined in Fig. 90 (a)-(f).

5.1.1.2 Substrate Network Settings

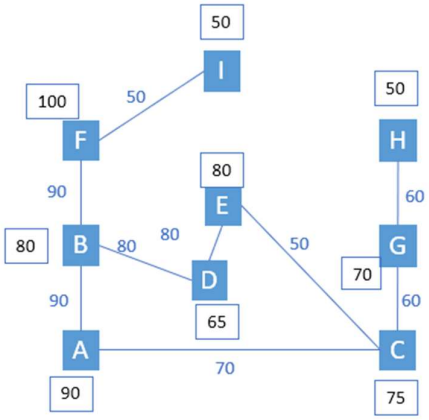


Fig.91 Sparse SN

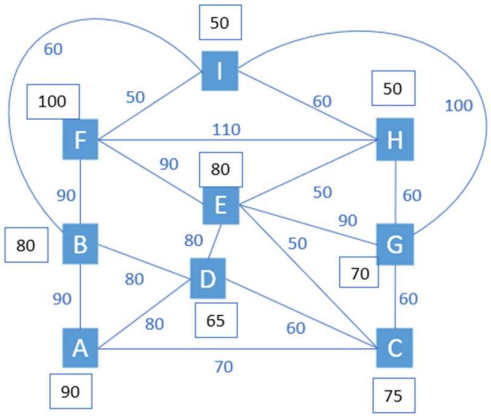


Fig.92 Semi-Dense SN

The SN used in the exhaustive numerical analysis is categorized into three distinct density levels, each representing a different level of connectivity and resource availability. Similar to the VN topology selection, the inclusion of all three SN density levels ensures a comprehensive evaluation of our methods under different resource

doi:10.6342/NTU202501591

constraints, validating whether they remain effective across various SN environments.

The Sparse SN (Fig. 91) has limited resources and low connectivity, meaning there are fewer available substrate links for mapping. This constraint makes it difficult to find efficient augmentation solutions. The Semi-Dense SN (Fig. 92) has moderate connectivity, providing a balanced test scenario where more substrate links are available but still constrained compared to dense networks. The Dense SN is derived from the semi-dense SN by adding extra substrate links to enhance connectivity. The additional links are represented using the format Link(Node1, Node2, Bandwidth), where Node1 and Node2 indicate the two endpoints of the substrate link, and Bandwidth represents the available bandwidth for that link.

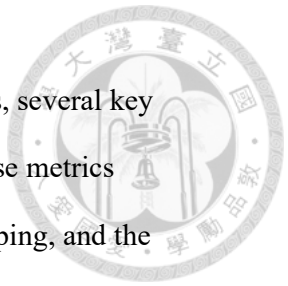
Link('A', 'E', 100), Link('A', 'F', 70), Link('A', 'G', 80), Link('A', 'H', 60), Link('A', 'I', 50),
Link('B', 'C', 100), Link('B', 'E', 60), Link('B', 'G', 80), Link('B', 'H', 110),
Link('C', 'F', 60), Link('C', 'H', 80), Link('C', 'I', 90),
Link('D', 'F', 90), Link('D', 'G', 110), Link('D', 'H', 50), Link('D', 'I', 60),
Link('E', 'I', 100), Link('F', 'G', 80),

These additional substrate links offer abundant resources, allowing survivability with minimal additional cost.

By testing the proposed methods across these three SN types, we evaluate their effectiveness in different network environments, ensuring robustness across various substrate resource conditions.

doi:10.6342/NTU202501591

5.1.1.3 Performance Metrics



To evaluate the effectiveness of the augmentation and mapping methods, several key performance metrics are used in the exhaustive numerical analysis. These metrics assess the quality of the augmented VN selection, the efficiency of mapping, and the linearity of the mapping algorithm.

Increment Difference (Δ_{VN}) represents the difference between the increment of the augmented VN selected by the algorithm and the increment of the optimal augmented VN under a given augmentation strategy. This metric reflects the effectiveness of the algorithm in identifying the minimal increment augmented VN.

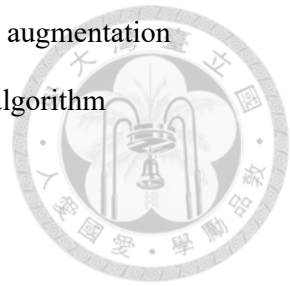
Mapping Cost Difference (Δ_{SN}) quantifies the difference between the mapping cost of the algorithm-selected augmented VN and the lowest mapping cost obtained among all augmented VNs under the same augmentation strategy. This metric evaluates the overall performance of the designed algorithm, including both the algorithm for selecting the augmented VN and the mapping algorithm.

Correlation Coefficient (ρ) measures the linearity between the increment and the mapping cost across all augmented VNs under a given augmentation strategy. Since correlation coefficient quantifies the degree of linearity, this metric indicates the extent to which the mapping algorithm exhibits linear characteristics.

Furthermore, we define two key mapping cost indicators: $\mathbf{cost_{opt}}$ and $\mathbf{cost_{heu}}$. The term $\mathbf{cost_{opt}}$ represents the lowest mapping cost obtained among all augmented VNs under a given augmentation strategy, serving as the optimal reference.

doi:10.6342/NTU202501591

The term $\mathbf{cost}_{heu}$ represents the mapping cost when the algorithm-based augmentation method selects an augmented VN, providing insight into how well the algorithm approximates the optimal mapping cost.



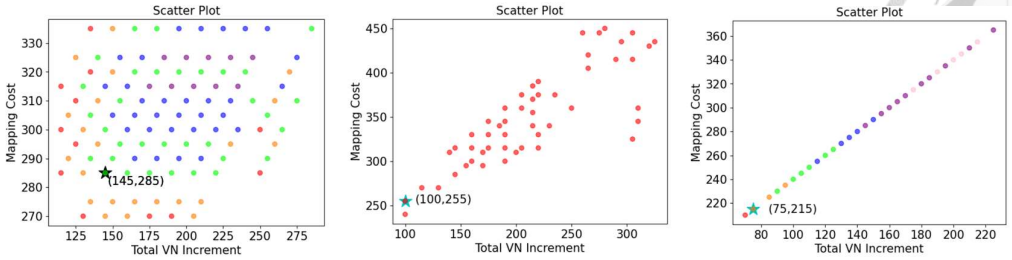
These performance metrics collectively provide a comprehensive evaluation of the augmentation and mapping strategies, ensuring that the proposed method is effective across various SN conditions and VN topologies.

To illustrate the distribution of augmented VNs in the exhaustive numerical analysis results, data points in the figures are color-coded based on the number of augmented VNs sharing the same increment and mapping cost. Specifically, red, orange, green, blue, purple, and pink represent 1–9, 10–99, 100–999, 1,000–9,999, 10,000–99,999, and 100,000 or more augmented VNs, respectively.

Additionally, in the scatter plots under SUDN, solutions found using the FIP method from [16] are marked with black stars, and solutions found using the heuristic under the FDP method from [16] are marked with blue-green stars. In the scatter plots under MRPL, solutions identified by the ProRed method from [17] are marked with blue-green stars. Similarly, in the scatter plots under MUPL, solutions obtained using our Nest-Base algorithm are also marked with blue-green stars.

doi:10.6342/NTU202501591

5.1.2 Results in Sparse Substrate Network



(a) (b) (c)

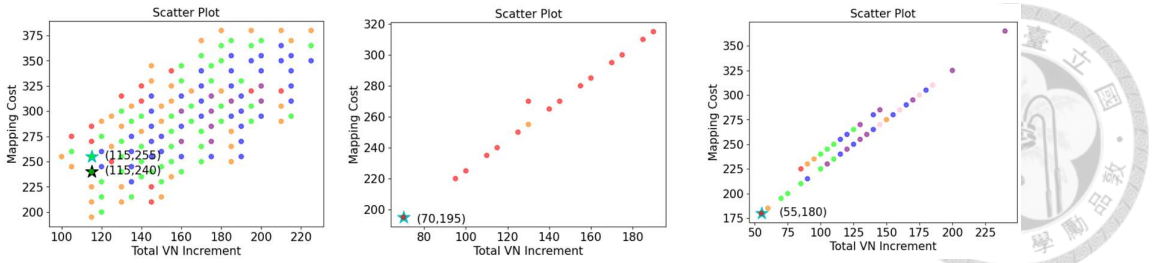
Fig. 93 Exhaustive Numerical Analysis Results in Sparse SN Using Line VN Topology:

(a) SUDN. (b) MRPL. (c) MUPL.

Line	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	270	240	210
$cost_{heu}$	FIP:285 FDP:285	255	215
Δ_{VN}	FIP: 145-115=30(26.1%) FDP:145-115=30(26.1%)	100-100=0(0%)	75-70=5(7.14%)
Δ_{SN}	FIP: 285-270=15(5.6%) FDP:285-270=15(5.6%)	255-240=15(6.25%)	215-210=5(2.38%)
ρ of all data	0.29042	0.841489	1

Table 5 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Sparse SN Using Line VN

doi:10.6342/NTU202501591



(a) (b) (c)

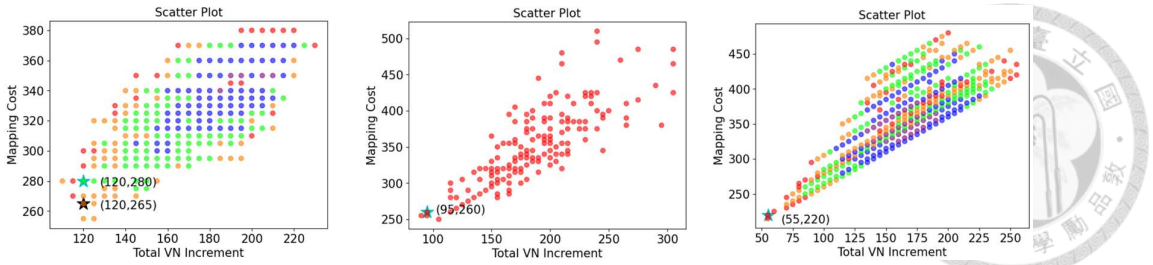
Fig. 94 Exhaustive Numerical Analysis Results in Sparse SN Using Star VN Topology:

(a) SUDN. (b) MRPL. (c) MUPL.

Star	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	195	195	180
$cost_{heu}$	FIP:240 FDP:255	195	180
Δ_{VN}	FIP: 115-100=15(15%) FDP:115-100=15(15%)	70-70=0(0%)	55-55=0(0%)
Δ_{SN}	FIP: 240-195=45(23.1%) FDP:255-195=60(30.8%)	195-195=0(0%)	180-180=0(0%)
ρ of all data	0.689688	0.993823	0.992763

Table 6 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Sparse SN Using Star VN

doi:10.6342/NTU202501591



(a) (b) (c)

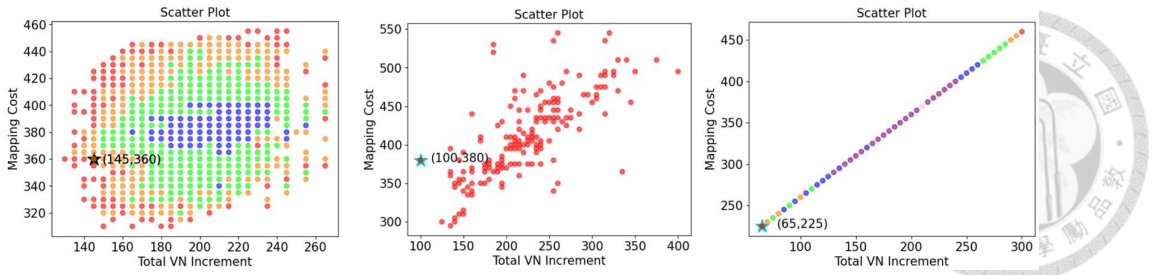
Fig. 95 Exhaustive Numerical Analysis Results in Sparse SN Using Star+1 VN Topology:

(a) SUDN. (b) MRPL. (c) MUPL.

Star+1	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	255	250	215
$cost_{heu}$	FIP:265 FDP:280	260	220
Δ_{VN}	FIP: 120-110=10(9%) FDP:120-110=10(9%)	95-90=5(5.6%)	55-55=0(0%)
Δ_{SN}	FIP: 265-255=10(3.9%) FDP:280-255=25(9.8%)	260-250=10(4%)	220-215=5(2.33%)
ρ of all data	0.698841	0.82309	0.656854

Table 7 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Sparse SN Using Star+1 VN

doi:10.6342/NTU202501591



(a) (b) (c)

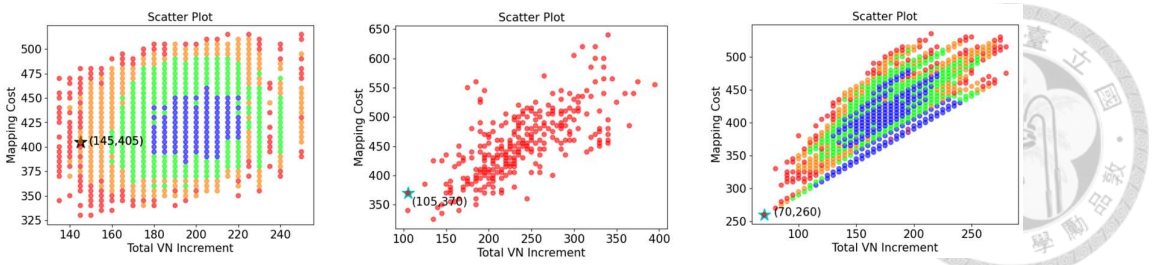
Fig. 96 Exhaustive Numerical Analysis Results in Sparse SN Using Ring VN Topology:

(a) SUDN. (b) MRPL. (c) MUPL.

Ring	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	310	295	225
$cost_{heu}$	FIP:360 FDP:360	380	225
Δ_{VN}	FIP: 145-135=10(7.4%) FDP:145-135=10(7.4%)	100-100=0(0%)	65-65=0(0%)
Δ_{SN}	FIP: 360-310=50(16.1%) FDP:360-310=50(16.1%)	380-295=85(28.8%)	225-225=0(0%)
ρ of all data	0.181108	0.76555	1

Table 8 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Sparse SN Using Ring VN

doi:10.6342/NTU202501591



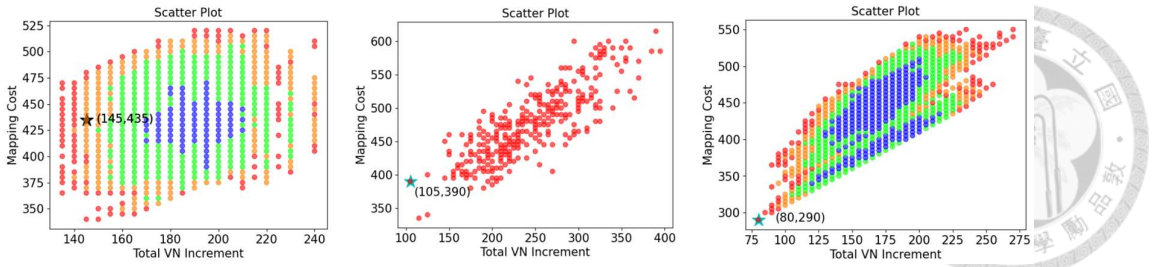
(a) (b) (c)

Fig. 97 Exhaustive Numerical Analysis Results in Sparse SN Using Fully Connected-1 VN Topology:(a) SUDN. (b) MRPL. (c) MUPL.

Fully Connected-1	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	330	325	260
$cost_{heu}$	FIP:405 FDP:405	370	260
Δ_{VN}	FIP: 145-135=10(7.4%) FDP:145-135=10(7.4%)	105-105=0(0%)	70-70=0(0%)
Δ_{SN}	FIP: 405-330=75(22.7%) FDP:405-330=75(22.7%)	370-325=45(13.8%)	260-260=0(0%)
ρ of all data	0.141915	0.708629	0.611217

Table 9 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Sparse SN Using Fully Connected-1VN

doi:10.6342/NTU202501591



(a) (b) (c)

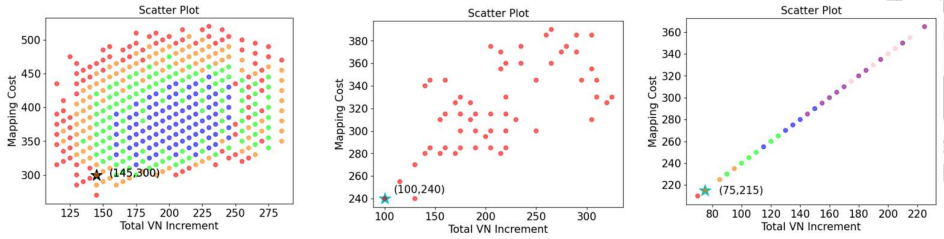
Fig. 98 Exhaustive Numerical Analysis Results in Sparse SN Using Fully Connected VN Topology:(a) SUDN. (b) MRPL. (c) MUPL.

Fully Connected	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	340	335	290
$cost_{heu}$	FIP:435 FDP:435	390	290
Δ_{VN}	FIP: 145-135=10(7.4%) FDP:145-135=10(7.4%)	105-105=0(0%)	80-80=0(0%)
Δ_{SN}	FIP: 435-340=95(27.9%) FDP:435-340=95(27.9%)	390-335=55(16.4%)	290-290=0(0%)
ρ of all data	0.105457	0.825677	0.605601

Table 10 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Sparse SN Using Fully Connected VN

doi:10.6342/NTU202501591

5.1.3 Results in Semi-Dense Substrate Network



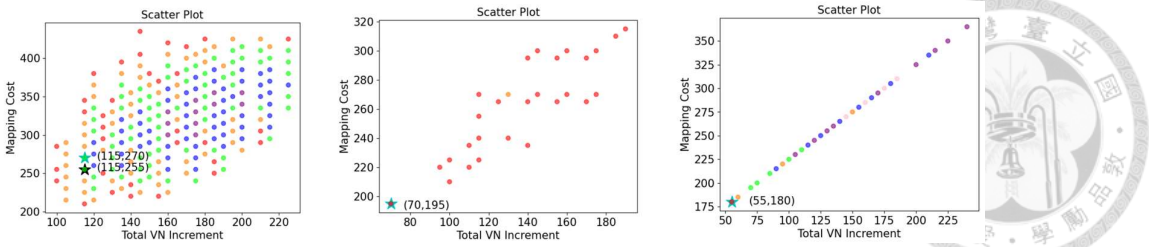
(a) (b) (c)

Fig. 99 Exhaustive Numerical Analysis Results in Semi-Dense SN Using Line VN Topology:(a) SUDN. (b) MRPL. (c) MUPL.

Line	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	270	240	210
$cost_{heu}$	FIP:300 FDP:300	240	215
Δ_{VN}	FIP: 145-115=30(26.1%) FDP:145-115=30(26.1%)	100-100=0(0%)	75-70=5(7.14%)
Δ_{SN}	FIP: 300-270=30(11.1%) FDP:300-270=30(11.1%)	240-240=0(0%)	215-210=5(2.38%)
ρ of all data	0.285561	0.682566	1

Table 11 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Semi-Dense SN Using Line VN

doi:10.6342/NTU202501591



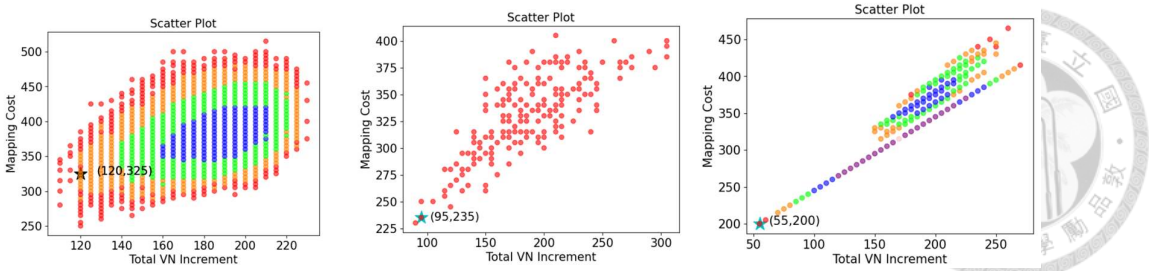
(a) (b) (c)

Fig. 100 Exhaustive Numerical Analysis Results in Semi-Dense SN Using Star VN Topology:(a) SUDN. (b) MRPL. (c) MUPL.

Star	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	210	195	180
$cost_{heu}$	FIP:255 FDP:270	195	180
Δ_{VN}	FIP: 115-100=15(15%) FDP:115-100=15(15%)	70-70=0(0%)	55-55=0(0%)
Δ_{SN}	FIP: 255-210=45(21.4%) FDP:270-210=60(28.6%)	195-195=0(0%)	180-180=0(0%)
ρ of all data	0.565602	0.892022	1

Table 12 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Semi-Dense SN Using Star VN

doi:10.6342/NTU202501591



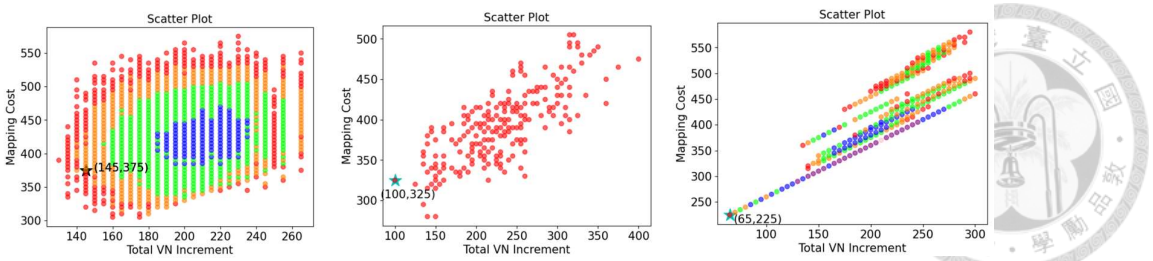
(a) (b) (c)

Fig. 101 Exhaustive Numerical Analysis Results in Semi-Dense SN Using Star+1 VN Topology:(a) SUDN. (b) MRPL. (c) MUPL.

Star+1	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	250	230	200
$cost_{heu}$	FIP:325 FDP:325	235	200
Δ_{VN}	FIP: 120-110=10(9%) FDP:120-110=10(9%)	95-90=5(5.6%)	55-55=0(0%)
Δ_{SN}	FIP: 325-250=75(30%) FDP:325-250=75(30%)	235-230=5(2.2%)	200-200=0(0%)
ρ of all data	0.435084	0.791947	0.954046

Table 13 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Semi-Dense SN Using Star+1 VN

doi:10.6342/NTU202501591



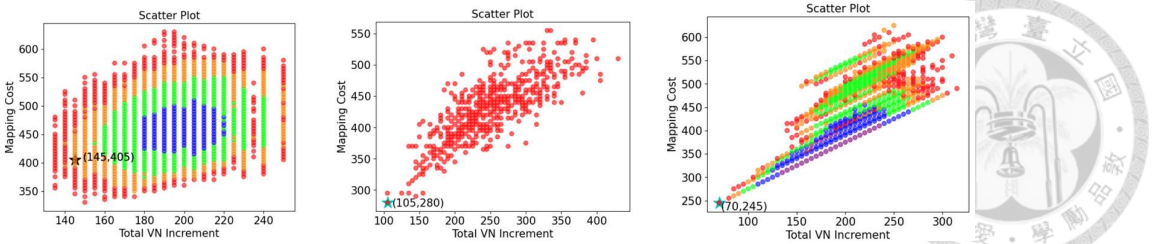
(a) (b) (c)

Fig. 102 Exhaustive Numerical Analysis Results in Semi-Dense SN Using Ring VN Topology:(a) SUDN. (b) MRPL. (c) MUPL.

Ring	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	305	280	225
$cost_{heu}$	FIP:375 FDP:375	325	225
Δ_{VN}	FIP: 145-130=15(11.5%) FDP:145-130=15(11.5%)	100-100=0(0%)	65-65=0(0%)
Δ_{SN}	FIP: 375-305=70(23%) FDP:375-305=70(23%)	325-280=45(16.1%)	225-225=0(0%)
ρ of all data	0.22256	0.762282	0.919592

Table 14 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Semi-Dense SN Using Ring VN

doi:10.6342/NTU202501591



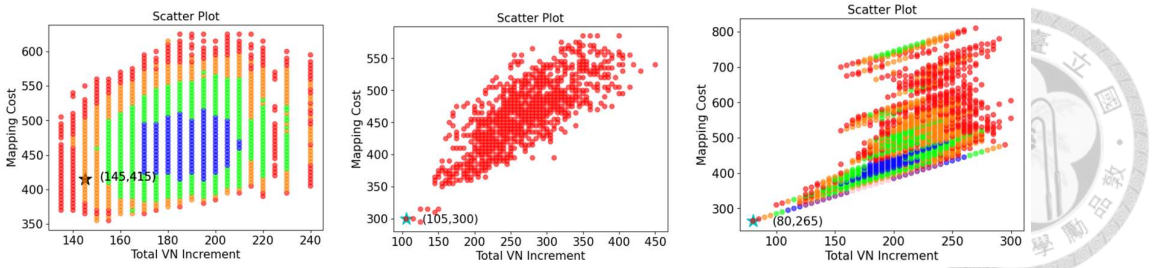
(a) (b) (c)

Fig. 103 Exhaustive Numerical Analysis Results in Semi-Dense SN Using Fully Connected-1 VN Topology:(a) SUDN. (b) MRPL. (c) MUPL.

Fully Connected-1	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	330	280	245
$cost_{heu}$	FIP:405 FDP:405	280	245
Δ_{VN}	FIP: 145-135=10(7.4%) FDP:145-135=10(7.4%)	105-105=0(0%)	70-70=0(0%)
Δ_{SN}	FIP: 405-330=75(22.7%) FDP:405-330=75(22.7%)	280-280=0(0%)	245-245=0(0%)
ρ of all data	0.233813	0.795033	0.814398

Table 15 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Semi-Dense SN Using Fully Connected-1VN

doi:10.6342/NTU202501591



(a) (b) (c)

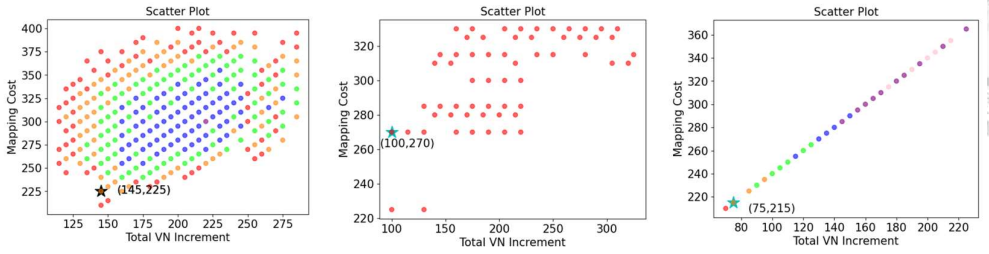
Fig. 104 Exhaustive Numerical Analysis Results in Semi-Dense SN Using Fully Connected VN Topology:(a) SUDN. (b) MRPL. (c) MUPL.

Fully Connected	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	355	295	265
$cost_{heu}$	FIP:415 FDP:415	300	265
Δ_{VN}	FIP: 145-135=10(7.4%) FDP:145-135=10(7.4%)	105-105=0(0%)	80-80=0(0%)
Δ_{SN}	FIP: 415-355=60(16.9%) FDP:415-355=60(16.9%)	300-295=5(1.7%)	265-265=0(0%)
ρ of all data	0.169231	0.785948	0.78327

Table 16 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Semi-Dense SN Using Fully Connected VN

doi:10.6342/NTU202501591

5.1.4 Results in Dense Substrate Network



(a) (b) (c)

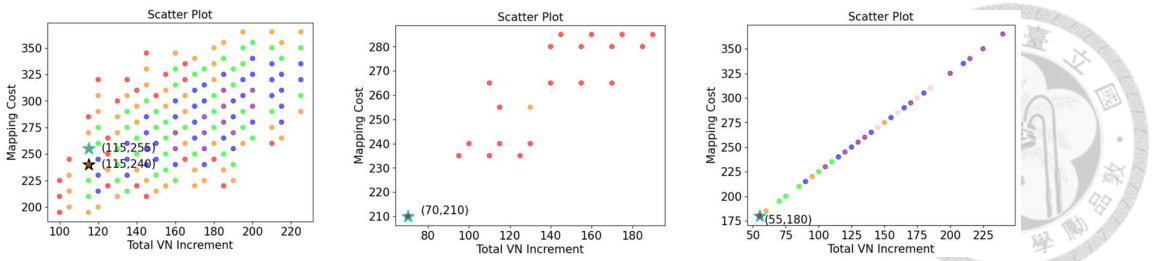
Fig. 105 Exhaustive Numerical Analysis Results in Dense SN Using Line VN Topology:

(a) SUDN. (b) MRPL. (c) MUPL.

Line	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	210	225	210
$cost_{heu}$	FIP:225 FDP:225	270	215
Δ_{VN}	FIP: 145-115=30(26.1%) FDP:145-115=30(26.1%)	100-100=0(0%)	75-70=5(7.14%)
Δ_{SN}	FIP: 225-210=15(7.1%) FDP:225-210=15(7.1%)	270-225=45(20%)	215-210=5(2.38%)
ρ of all data	0.446264	0.610093	1

Table 17 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Dense SN Using Line VN

doi:10.6342/NTU202501591



(a) (b) (c)

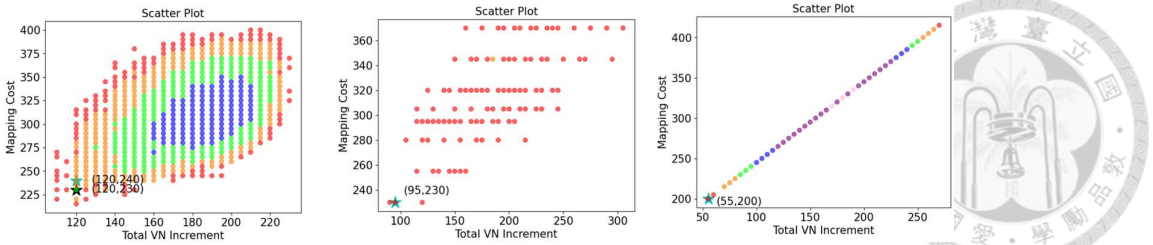
Fig. 106 Exhaustive Numerical Analysis Results in Dense SN Using Star VN Topology:

(a) SUDN. (b) MRPL. (c) MUPL.

Star	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	195	210	180
$cost_{heu}$	FIP:240 FDP:255	210	180
Δ_{VN}	FIP: 115-100=15(15%) FDP:115-100=15(15%)	70-70=0(0%)	55-55=0(0%)
Δ_{SN}	FIP: 210-195=15(7.7%) FDP:225-195=30(15.4%)	210-210=0(0%)	180-180=0(0%)
ρ of all data	0.720194	0.888234	1

Table 18 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Dense SN Using Star VN

doi:10.6342/NTU202501591



(a) (b) (c)

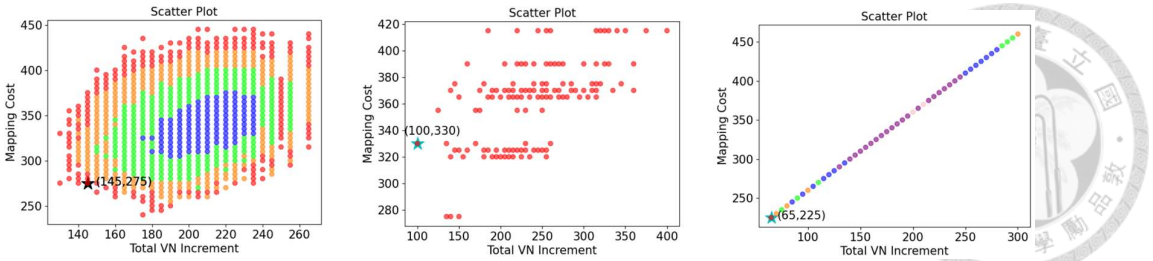
Fig. 107 Exhaustive Numerical Analysis Results in Dense SN Using Star+1 VN Topology:

(a) SUDN. (b) MRPL. (c) MUPL.

Star+1	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	215	230	200
$cost_{heu}$	FIP:230 FDP:240	230	200
Δ_{VN}	FIP: 120-110=10(9%) FDP:120-110=10(9%)	95-90=5(5.6%)	55-55=0(0%)
Δ_{SN}	FIP: 230-215=15(7%) FDP:240-215=25(11.6%)	230-230=0(0%)	200-200=0(0%)
ρ of all data	0.505932	0.666945	1

Table 19 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Dense SN Using Star+1 VN

doi:10.6342/NTU202501591



(a) (b) (c)

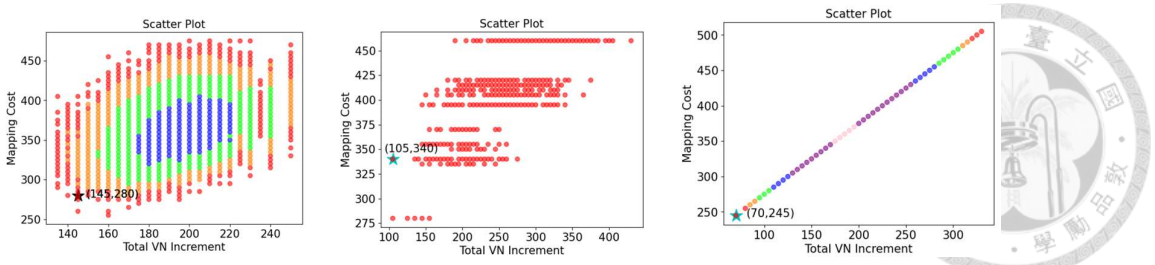
Fig. 108 Exhaustive Numerical Analysis Results in Dense SN Using Ring VN Topology:

(a) SUDN. (b) MRPL. (c) MUPL.

Ring	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	240	275	225
$cost_{heu}$	FIP:275 FDP:275	330	225
Δ_{VN}	FIP: 145-130=15(11.5%) FDP:145-130=15(11.5%)	100-100=0(0%)	65-65=0(0%)
Δ_{SN}	FIP: 275-240=35(14.6%) FDP:275-240=35(14.6%)	330-275=55(20%)	225-225=0(0%)
ρ of all data	0.382878	0.518433	1

Table 20 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Dense SN Using Ring VN

doi:10.6342/NTU202501591



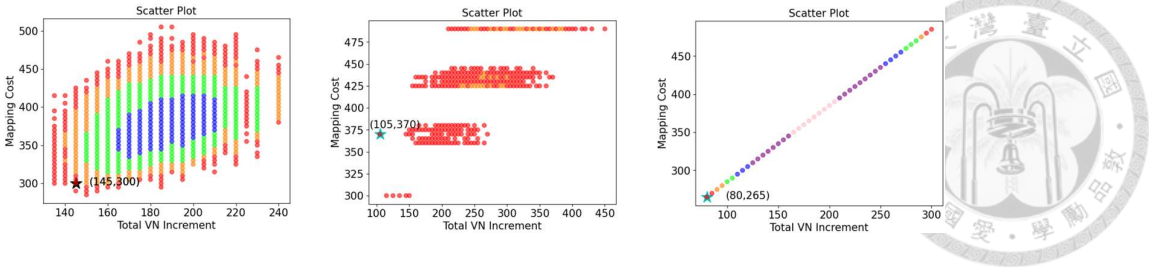
(a) (b) (c)

Fig. 109 Exhaustive Numerical Analysis Results in Dense SN Using Fully Connected-1 VN Topology:(a) SUDN. (b) MRPL. (c) MUPL.

Fully Connected-1	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	255	280	245
$cost_{heu}$	FIP:280 FDP:280	340	245
Δ_{VN}	FIP: 145-135=10(7.4%) FDP:145-135=10(7.4%)	105-105=0(0%)	70-70=0(0%)
Δ_{SN}	FIP: 280-255=25(9.8%) FDP:280-255=25(9.8%)	340-280=60(21.4%)	245-245=0(0%)
ρ of all data	0.371448	0.617297	1

Table 21 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Dense SN Using Fully Connected-1 VN

doi:10.6342/NTU202501591



(a) (b) (c)

Fig. 110 Exhaustive Numerical Analysis Results in Dense SN Using Fully Connected VN Topology:(a) SUDN. (b) MRPL. (c) MUPL.

Fully Connected	SUDN (FDP in [16])	MRPL [17]	MUPL (Our strategy)
$cost_{opt}$	285	300	265
$cost_{heu}$	FIP:300 FDP:300	370	265
Δ_{VN}	FIP: 145-135=10(7.4%) FDP:145-135=10(7.4%)	105-105=0(0%)	80-80=0(0%)
Δ_{SN}	FIP: 300-285=15(5.26%) FDP:300-285=15(5.26%)	370-300=70(23.3%)	265-265=0(0%)
ρ of all data	0.422599	0.65616	1

Table 22 Heuristic Performance Analysis: Increment, Mapping Cost, and Correlation in Dense SN Using Fully Connected VN

doi:10.6342/NTU202501591

5.1.5 Summary and Key Observations

5.1.5.1 The Efficiency of the Proposed Algorithm in Finding the Globally Optimal Minimum-Increment Augmented VN

The heuristic in SUDN [16] struggles in simple VN topologies, leading to a significant gap between the found and optimal increments (e.g., 26.1% error in Line VN), but its accuracy improves in more complex VN topologies (e.g., 7.4% error in Fully Connected VN). Additionally, the heuristic in [16] fails to effectively identify the minimum increment augmented VN, as it consistently selects solutions with the same increment as FIP, indicating its inefficiency. This limitation arises because FDP evaluates backup resources separately for each failure case, rather than optimizing the backup resource configuration holistically, leading to suboptimal augmentation choices.

In contrast, the heuristic in MRPL [17] consistently finds the minimum increment augmented VN across all VN topologies due to its smaller solution space, making it highly efficient.

Our proposed algorithm achieves the same level of accuracy as [17] but operates under MUPL, which has a much larger solution space. Despite this increased complexity, our algorithm can still reliably find the globally optimal minimum-increment augmented VN in small-scale VNs, demonstrating its effectiveness even under significantly expanded augmentation constraints.

doi:10.6342/NTU202501591

5.1.5.2 Mapping (embedding) cost and Linearity of the Mapping Algorithm

The mapping algorithm in SUDN [16] exhibits poor linearity, especially in sparse SN, where the correlation coefficient is notably low except in Star and Star+1 VN topologies. This lack of linearity affects mapping cost, as seen in the Fully Connected-1 and Fully Connected VN topologies, where even though the augmentation VN increment deviates from the optimal by only 10 (7.4%), the mapping cost deviates significantly by 75 (22.7%) and 95 (27.9%). However, in dense SN, the correlation coefficient improves, and the mapping cost gap is reduced to 25 (9.8%) and 15 (5.26%), indicating that [16] performs better in dense SN.

The mapping algorithm in MRPL [17] exhibits a more linearity across all VN topologies, as observed in both correlation coefficients and scatter plots. However, despite its more linearity and accurate augmentation selection, its performance remains suboptimal in complex VN topologies. For Ring, Fully Connected-1, and Fully Connected VN, the mapping cost deviation from the optimal remains between 13.8% and 28.8%, suggesting that [17] is not well-suited for complex VN topologies. Additionally, in dense SN, even the simplest topology (Line VN) results in a 20% mapping cost deviation, indicating that [17] is also not well-suited for complex SN topologies.

Our mapping algorithm exhibits stronger linearity than both [16] and [17], as demonstrated by higher correlation coefficients in most cases. Even in cases where the correlation coefficient is relatively lower, the scatter plots show that our mapping algorithm still achieves the minimum mapping cost, ensuring that the mapping performance remains optimal despite minor deviations in linearity. These results

doi:10.6342/NTU202501591

ensure that across all VN topologies and all SN densities, the mapping cost deviation from the theoretical minimum remains within 5 (2.38%), indicating that our mapping algorithm performs optimally regardless of VN and SN topology complexity.

Notably, in dense SNs, our mapping algorithm achieves perfect linearity (correlation coefficient = 1), meaning that as the SN becomes denser, our algorithm better maintains the linear relationship between the augmented VN increment and mapping cost. This implies that our method can perfectly predict the impact of augmentation on mapping cost in dense SNs.

5.1.5.3 Relationship Between Solution Space Size and Minimum Mapping (Embedding) Cost

The results also reveal a correlation between augmentation solution space size and the ability to achieve lower minimum mapping costs. SUDN [16] has a significantly larger solution space than MRPL [17], yet MRPL achieves lower minimum mapping costs than SUDN in sparse and semi-dense SNs. However, in Fully Connected SN, SUDN achieves lower minimum mapping costs than MRPL, suggesting that a larger solution space is more beneficial when the SN topology is more complex. Conversely, when the SN topology is simpler, MRPL's augmentation strategy itself is inherently more suited for sparse SNs, allowing it to achieve lower minimum mapping costs than SUDN.

Since MUPL (Our strategy) is an extension of MRPL with relaxed node migration constraints, it retains MRPL's efficiency in smaller SNs while significantly expanding the augmentation solution space. This ensures that MUPL not only inherits MRPL's efficiency in sparse SN but also surpasses MRPL in sparse SN by leveraging a more

doi:10.6342/NTU202501591

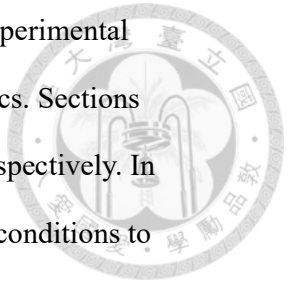
flexible augmentation space. Additionally, in dense SN, the larger solution space of MUPL enables it to outperform SUDN, achieving a more optimal augmentation configuration. As a result, our augmentation strategy consistently achieves the lowest minimum mapping costs across all SN topologies, regardless of whether the SN is sparse, semi-dense, or dense.

5.2 Performance Evaluation under Real-World Network Scenarios

This section presents the simulation-based evaluation of the proposed Nest-Base algorithm under realistic network scenarios. We compare the performance of Nest-Base with three existing augmentation and mapping methods: FIP and FDP from [16] and ProRed from [17]. To comprehensively evaluate the effectiveness of Nest-Base, we measure its performance based on two key metrics: mapping (embedding) cost and acceptance rate. The evaluation is conducted across different SN conditions, considering both small-scale SN and large-scale SN, both of which are designed to align with real-world network environments. Additionally, we consider different workload levels to assess the adaptability of each mapping algorithm. Under moderate load, the SN handles a reasonable number of VN requests, allowing for stable resource allocation and making it suitable for evaluating mapping (embedding) cost. Under high load, the SN experiences a higher number of VN requests, leading to resource limitation and making it an appropriate condition for evaluating acceptance rate, which measures how well different methods handle VN embedding when resources are constrained.

doi:10.6342/NTU202501591

The structure of this section is as follows. Section 5.2.1 describes the experimental setup, including the SN and VN configurations and the evaluation metrics. Sections 5.2.2 and 5.2.3 present the results for small-scale and large-scale SN, respectively. In both sections, simulations are conducted under moderate and high load conditions to evaluate mapping (embedding) cost and acceptance rate, respectively.



5.2.1 Experimental Setup

This section provides the details of the experimental setup used in real-world network scenarios. Section 5.2.1.1 describes the VN configurations, including different VN topologies and workload conditions to ensure a realistic evaluation. Section 5.2.1.2 outlines the SN configurations, covering different network sizes and levels of resource availability to simulate practical deployment environments. Section 5.2.1.3 defines the performance metrics, detailing the criteria used to assess mapping (embedding) cost and acceptance rates under various workload conditions.

5.2.1.1 Virtual Network Settings

The VN settings are defined to reflect different network conditions under moderate and high workload scenarios. Table 23 presents the VN settings under moderate load, while Table 24 details the VN settings under high load.

doi:10.6342/NTU202501591

Parameter	Value
type	Line topology Ring topology Random topology
Number of virtual nodes	U[2~10]
Computing and bandwidth demand	U[1~30]
Arrival rate of VNR	Poisson, 1 VNR/ per 10 unit time
Lifetime of VNR	Exponential distribution, 100 unit time/per VNR
Link connection probability (Random topology)	0.5

Table 23 VNs Settings under Real-World Network Scenarios (Moderate Workload)

Parameter	Value
type	Line topology Ring topology Random topology
Number of virtual nodes	U[2~10]
Computing demand	U[1~15]
Bandwidth demand	U[1~30]
Arrival rate of VNR	Poisson, 1 VNR/ per 10 unit time
Lifetime of VNR	Exponential distribution, 1000 unit time/per VNR
Link connection probability (Random topology)	0.5

Table 24 VNs Settings under Real-World Network Scenarios

(High Workload)

The VN topology can be line, ring, or random, ensuring diverse network structures. The line topology represents sequential processing applications, such as a face ID login application can be broken down into multiple stages—such as image capturing,

doi:10.6342/NTU202501591

compression, face detection, identity verification, and login—where each step depends on the output of the previous one. The ring topology is suitable for cyclic processing systems, such as IoT sensor data processing that continuously collects and transmits information in a loop. The random topology reflects the diversity of real-world network configurations.

The number of virtual nodes is randomly selected within a defined range, and the computing and bandwidth demands follow a uniform distribution. The VN request (VNR) arrival follows a Poisson process, with an average rate of one request per 10 unit time, while the VN lifetime follows an exponential distribution. In random topology scenarios, the probability of link formation between nodes is set at 0.5 to maintain network variability. The VN lifetime is set to 100 unit time per VNR under moderate load and 1000 unit time per VNR under high load, making the high-load scenario result in a significantly larger number of active VNs in the SN.

5.2.1.2 Substrate Network Settings

The SN settings are designed to reflect the infrastructures provided by real-world infrastructure providers (InPs), differentiating between small-scale and large-scale SN. Small-scale SN represent networks managed by small InPs, whereas large-scale SN correspond to networks managed by major InPs with more extensive resources. Table 25 presents the SN settings under moderate load, while Table 26 provides the SN settings under high load.

doi:10.6342/NTU202501591

Parameter	Value
Number of substrate nodes	Small:20 Large:50
Computing and bandwidth capacity	U[100~300]
Link connection probability	0.3~0.7

Table 25 SNs Settings under Real-World Network Scenarios

(Moderate Workload)

Parameter	Value
Number of substrate nodes	Small:20 Large:50
Computing and bandwidth capacity	U[100~300]
Link connection probability	Small:0.2 Large:0.1

Table 26 SNs Settings under Real-World Network Scenarios

(High Workload)

In Table 25, the link connection probability varies between 0.3 and 0.7, representing different levels of substrate links availability, from resource-constrained to resource-abundant environments. In contrast, Table 26 fixes the link connection probability at 0.2 for small SN and 0.1 for large SN, reflecting a high-load scenario where both the workload is high and substrate links availability is extremely scarce. This configuration ensures that the evaluation considers cases where SN experience both high demand and severe resource constraints.

doi:10.6342/NTU202501591

5.2.1.3 Performance Metrics

To evaluate the performance of the proposed method, two key metrics are used: mapping cost and acceptance rate. Under moderate load, SN resources are generally sufficient since VN requests do not heavily compete for resources. In this scenario, acceptance rates are typically 100%, making mapping cost the most suitable metric for evaluation.

Under extreme conditions where both high load and substrate links availability are extremely scarce, however, SN resources are heavily utilized since the SN resources are simultaneously utilized by many VN and overall resource scarcity. In such extreme conditions, the primary concern is no longer minimizing mapping cost but rather ensuring that more VNRs are successfully accepted, leading to VN successfully being embedded. As a result, acceptance rate is the key metric to assess how well algorithm can accommodate incoming VN requests in a highly constrained environment.

5.2.2 Results in Small Substrate Network

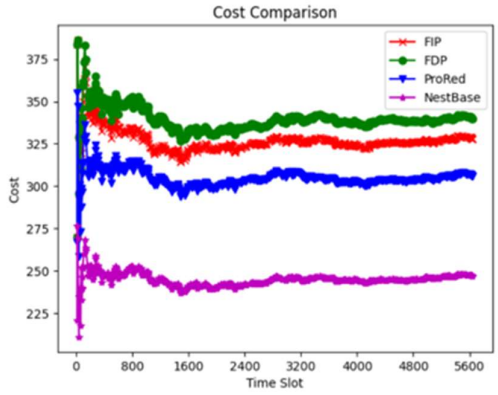
This section presents the simulation-based evaluation of the proposed augmentation and mapping algorithms, including the Nest-Base algorithm for augmented VN selection and the designed mapping algorithm under realistic network conditions. We compare the performance of Nest-Base with three existing augmentation and mapping methods: FIP and FDP from [16] and ProRed from [17].

The evaluation results in the small-scale SN environment are divided into two subsections. Section 5.2.2.1 evaluates the mapping cost under moderate load conditions to assess resource efficiency, while Section 5.2.2.2 analyzes the acceptance

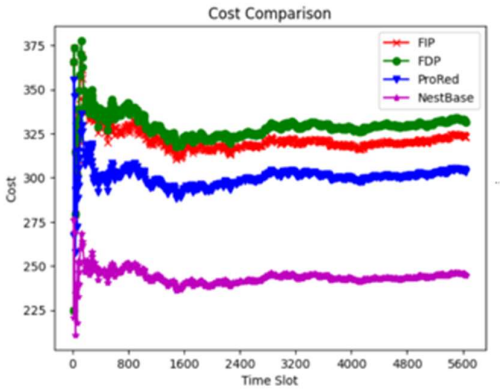
doi:10.6342/NTU202501591

rate under extreme conditions to measure the robustness of each method in handling VN requests when SN resources are severely constrained.

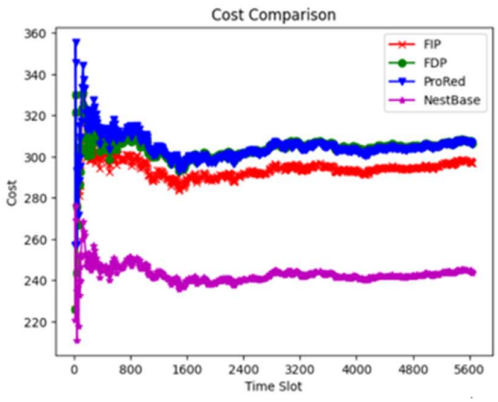
5.2.2.1 Mapping Cost Evaluation



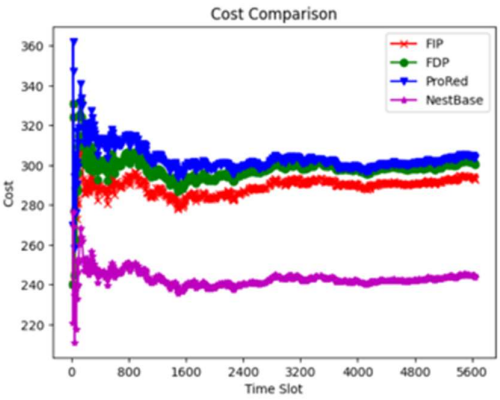
(a)



(b)

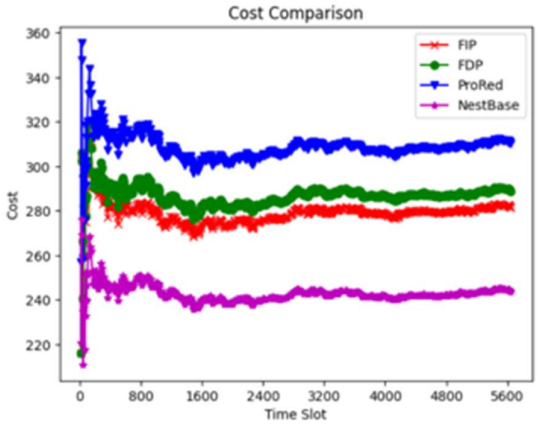


(c)



(d)

doi:10.6342/NTU202501591



(e)

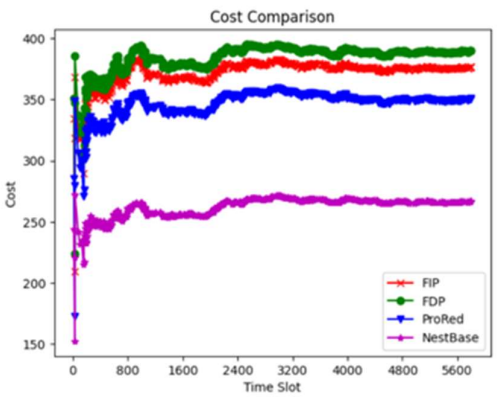
Fig.111 Average mapping (embedding) cost over time in Small SN with Line VNs

(a)-(e): SN Link Probability 0.3-0.7

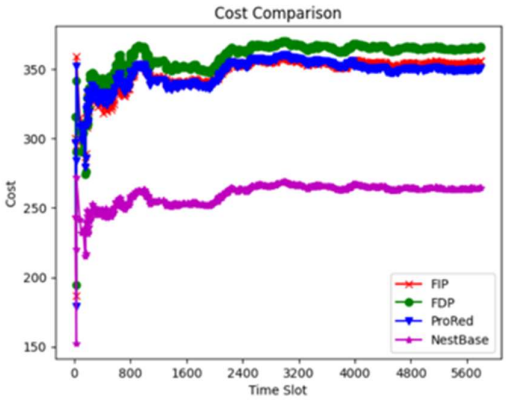
Line	0.3	0.4	0.5	0.6	0.7
FIP [16]	328	323	297	293	282
FDP [16]	340	332	306	301	289
ProRed [17]	306	303	306	303	310
NestBase	247	245	244	244	244

Table 27 Average mapping (embedding) cost in Small SN with Line VNs: SN Link Probability 0.3-0.7

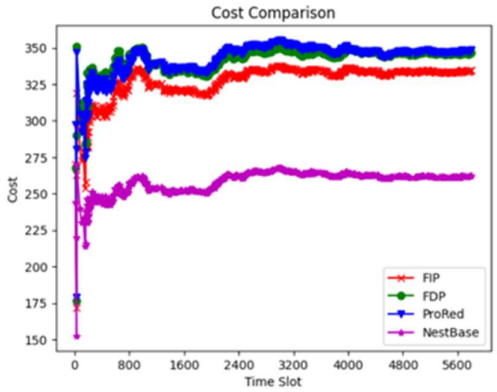
doi:10.6342/NTU202501591



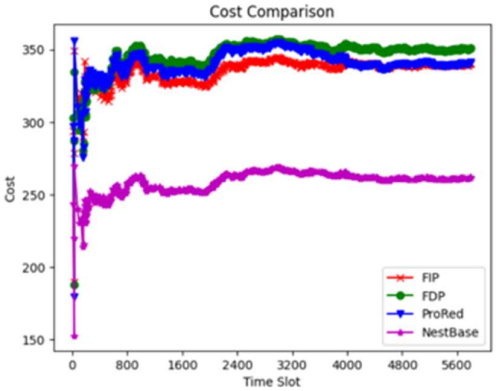
(a)



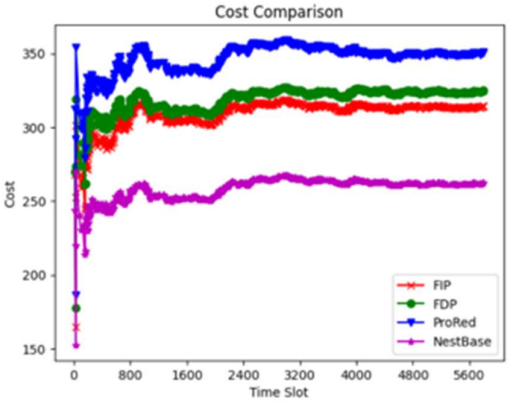
(b)



(c)



(d)



(e)

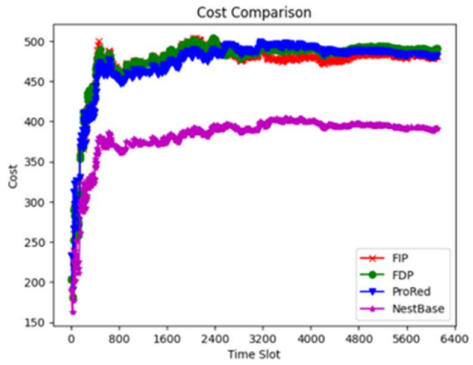
Fig.112 Average mapping (embedding) cost over time in Small SN with Ring VNs

(a)-(e): SN Link Probability 0.3-0.7

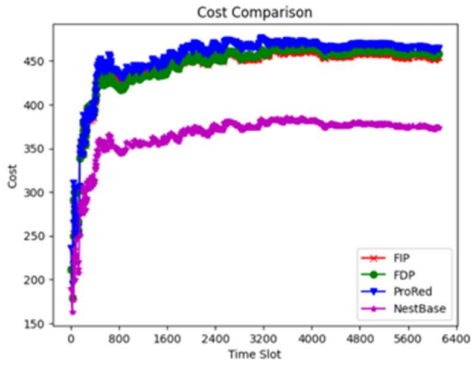
doi:10.6342/NTU202501591

Ring	0.3	0.4	0.5	0.6	0.7
FIP [16]	377	356	335	340	314
FDP [16]	390	366	347	351	325
ProRed [17]	350	351	348	341	351
NestBase	267	265	262	262	262

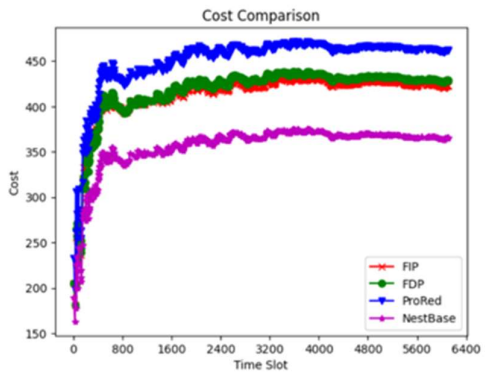
Table 28 Average mapping (embedding) cost in Small SN with Ring VNs: SN Link Probability 0.3-0.7



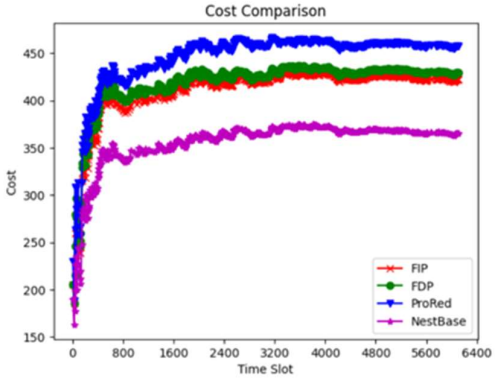
(a)



(b)

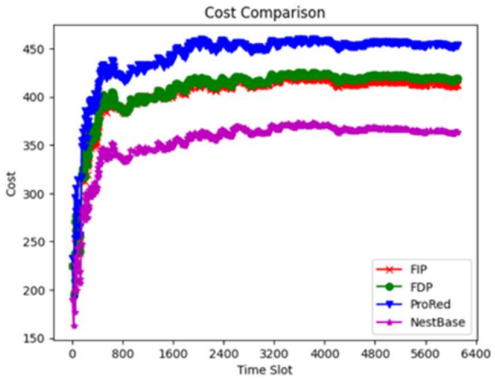


(c)



(d)

doi:10.6342/NTU202501591



(e)

Fig.113 Average mapping (embedding) cost over time in Small SN with Random VNs (a)-(e): SN Link Probability 0.3-0.7

Random	0.3	0.4	0.5	0.6	0.7
FIP [16]	482	455	423	422	413
FDP [16]	491	458	429	429	419
ProRed [17]	482	465	462	457	453
NestBase	391	374	365	365	365

Table 29 Average mapping (embedding) cost in Small SN with Random VNs: SN Link Probability 0.3-0.7

For [16], it is observed that the FDP method consistently results in higher mapping costs compared to FIP. We attribute this to the fact that the heuristic used in [16] for finding the augmented VN is unable to identify an augmentation with a smaller increment than FIP. As discussed in Section 2.3.1, this limitation arises because FDP evaluates backup resources separately for each failure scenario, rather than optimizing the backup resource allocation as a whole, leading to suboptimal augmentation decisions. Additionally, while the mapping algorithm in [16] does not exhibit strong linearity, it still maintains a positive correlation between augmentation increment and

doi:10.6342/NTU202501591

mapping cost, as confirmed in Section 5.1. Consequently, since FDP systematically selects augmented VNs with larger increments than FIP, this trend also translates into higher mapping costs for FDP compared to FIP in the long run.

For ProRed in [17], it is observed that its performance improvement over FIP is the most significant when the SN link probability is 0.3, while its performance is the worst compared to FIP when the SN link probability is 0.7. This simulation result further confirms that the method in ProRed performs significantly better in sparse SNs than in dense SNs, reinforcing the observation that ProRed is only suitable for sparse SN environments. As the SN becomes denser, its effectiveness deteriorates, leading to higher mapping costs compared to FIP. This finding is consistent with the conclusions drawn in Section 5.1. Additionally, the simulation results further confirm that ProRed performs better relative to FIP in simple VN topologies (e.g., Line VN) than in complex VN topologies (e.g., Random VN). This indicates that ProRed is only suitable for handling simple VN topologies, while its performance degrades when VN topologies become more complex, aligning with the conclusions of Section 5.1.

Our Nest-Base method consistently outperforms both [16] and [17] across all SN link probabilities and VN topologies, demonstrating its superior adaptability and efficiency in various network conditions.

In Line VN, our method achieves 25%–13% lower mapping costs compared to FIP when the SN link probability ranges from 0.3 to 0.7. Additionally, across all SN link probabilities, Nest-Base consistently performs 20% better than ProRed. In Ring VN,

doi:10.6342/NTU202501591

the Nest-Base algorithm achieves 31%–15% lower mapping costs than FIP for SN link probabilities between 0.3 and 0.7. Furthermore, it consistently outperforms ProRed by 24% across all SN link probabilities. In Random VN, our method achieves 19%–12% lower mapping costs than FIP across SN link probabilities from 0.3 to 0.7, while maintaining a 20% advantage over ProRed across all SN link probabilities.

These results confirm that our Nest-Base algorithm is not only effective in sparse SNs and simple VN topologies but also maintains superior performance in dense SNs and complex VN topologies, unlike [17], which struggle under such conditions.

5.2.2.2 Acceptance Rate Evaluation

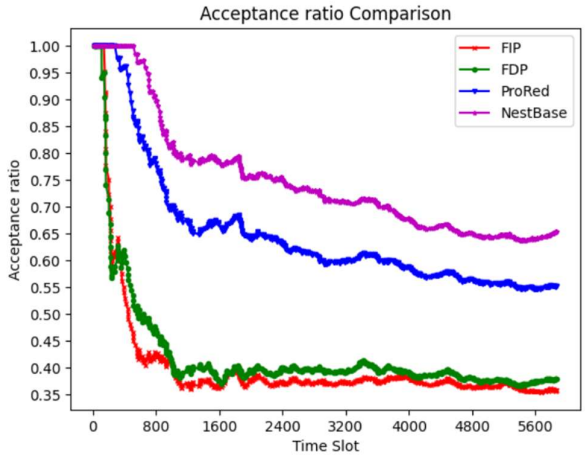


Fig.114 Comparison of VN acceptance ratio in Small SN with Line VNs

doi:10.6342/NTU202501591

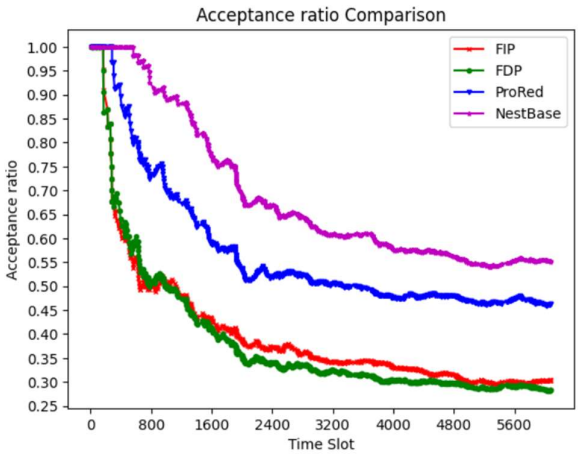


Fig.115 Comparison of VN acceptance ratio in Small SN with Ring VNs

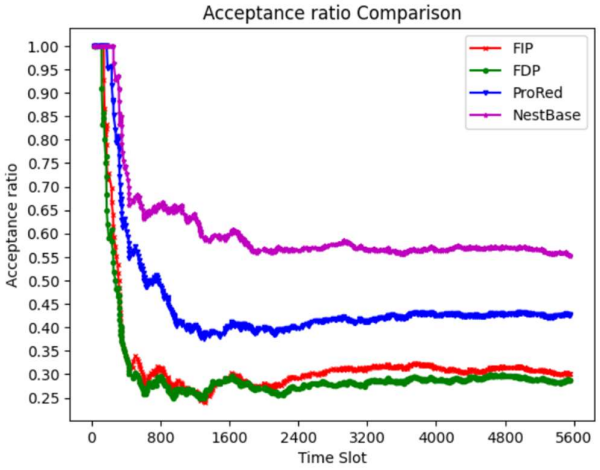


Fig.116 Comparison of VN acceptance ratio in Small SN with Random VNs

In comparison to moderate workload conditions, the extreme case of high workload not only imposes a tenfold increase in workload but also results in a significantly more resource-constrained SN, making the SN effectively sparser.

doi:10.6342/NTU202501591

Under these high-load conditions, [16] exhibits an acceptance rate of only 0.3–0.4, indicating that its augmentation and mapping strategy struggles to accommodate VNRs when SN resources are severely limited. Furthermore, based on the observations from Section 5.2.2.1, it is evident that while FDP generally performs worse than FIP, the difference is not substantial. As a result, both the FIP and FDP methods exhibit similar acceptance rates, consistently within the range of 0.3–0.4 under high-load conditions.

Based on the results from Section 5.1 and Section 5.2.2.1, ProRed demonstrates a stronger capability to operate in sparse SN environments compared to the FIP and FDP methods in [16]. Given that high-load conditions further exacerbate SN resource scarcity, ProRed achieves a higher acceptance rate than [16] under these conditions, confirming its advantage in sparse SNs. However, it is also observed that ProRed's acceptance rate varies depending on the VN topology, with a 0.6 acceptance rate for Line VN and a lower 0.4 acceptance rate for Random VN. This aligns with previous findings that ProRed struggles to handle complex VN topologies, leading to lower acceptance rates for more complex VN structures compared to simpler ones.

Our Nest-Base algorithm achieves a higher acceptance rate than both [16] and [17] under high-load conditions, demonstrating its superior ability to handle VNRs even when SN resources are severely constrained. Specifically, the acceptance rates for Line VN, Ring VN, and Random VN are 0.7, 0.6, and 0.6, respectively, indicating that our method maintains a relatively stable acceptance rate across different VN topologies.

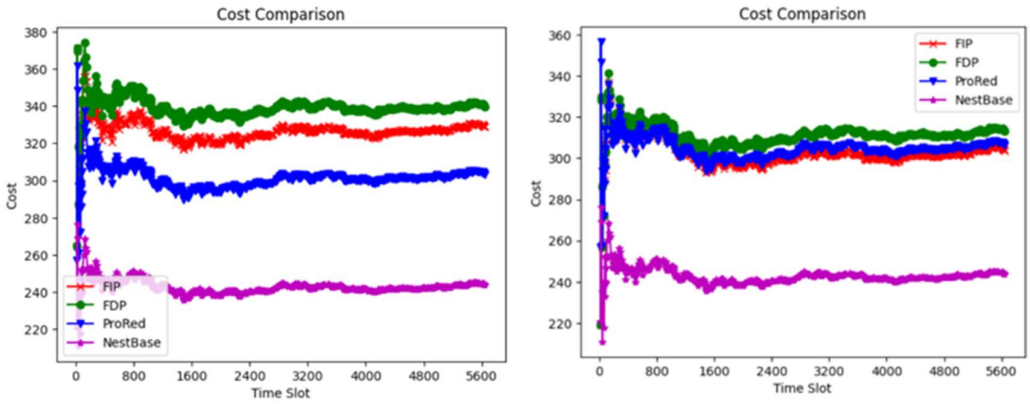
doi:10.6342/NTU202501591

Furthermore, the acceptance rate for Random VN is only 0.1 lower than that of Line VN, suggesting that Nest-Base remains stable and effective in handling complex VN topologies under high-load conditions. Additionally, even in small SNs with an SN link probability as low as 0.2, our method maintains an acceptance rate above 0.6, highlighting its robustness in extremely constrained network environments.

5.2.3 Results in Large Substrate Network

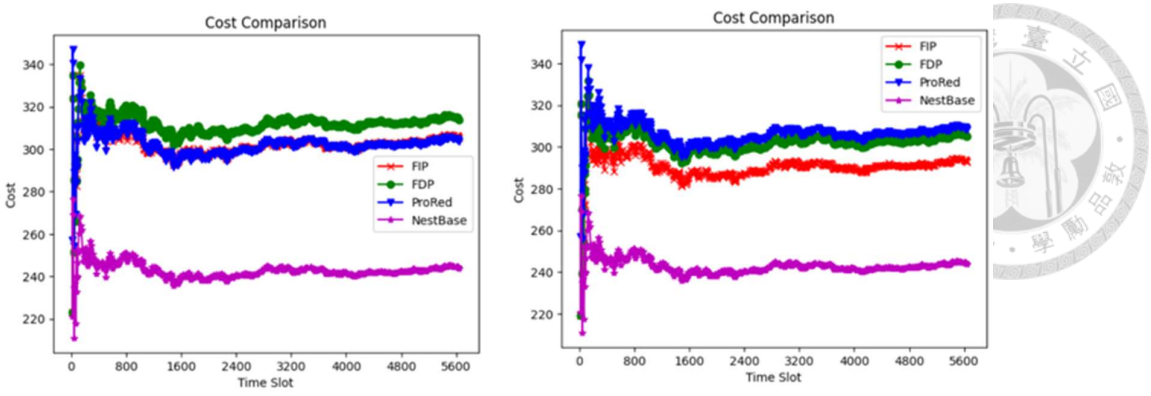
We compare the performance of Nest-Base with three existing augmentation and mapping methods: FIP and FDP from [16] and ProRed from [17]. The evaluation in the large-scale SN environment follows the same methodology as Section 5.2.2, with Section 5.2.3.1 assessing mapping cost under moderate load conditions and Section 5.2.3.2 analyzing acceptance rate under extreme conditions.

5.2.3.1 Mapping Cost Evaluation

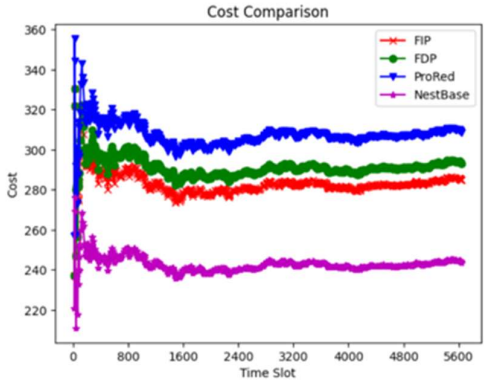


(a) (b)

doi:10.6342/NTU202501591



(c) (d)



(e)

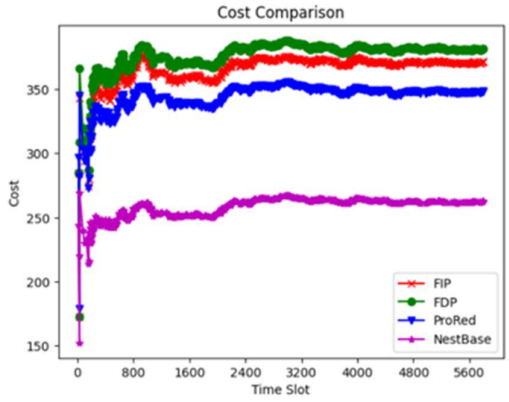
Fig.117 Average mapping (embedding) cost over time in Large SN with Line VNs

(a)-(e): SN Link Probability 0.3-0.7

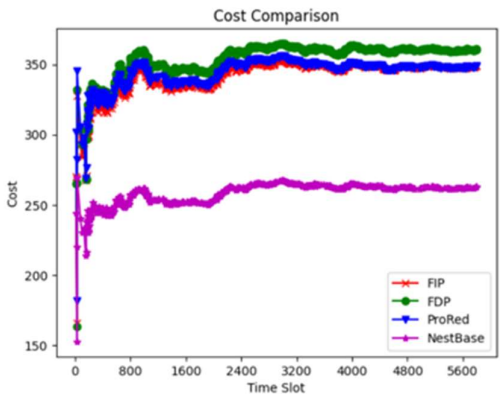
Line	0.3	0.4	0.5	0.6	0.7
FIP [16]	329	304	305	293	285
FDP [16]	340	313	314	305	293
ProRed [17]	303	306	304	308	309
NestBase	244	244	244	244	244

Table 30 Average mapping (embedding) cost in Large SN with Line VNs: SN Link Probability 0.3-0.7

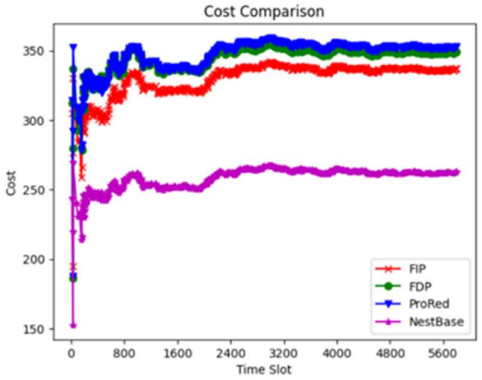
doi:10.6342/NTU202501591



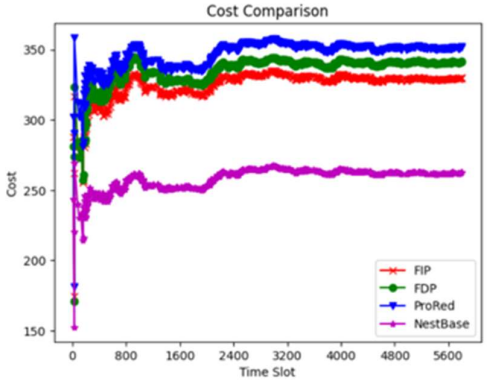
(a)



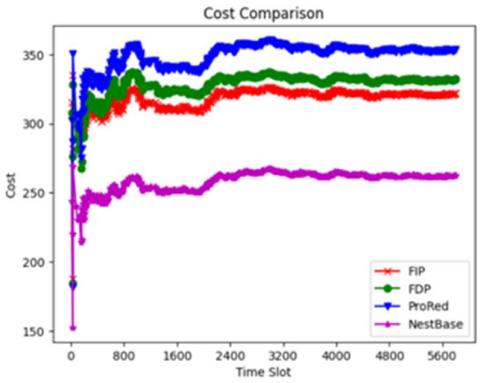
(b)



(c)



(d)



(e)

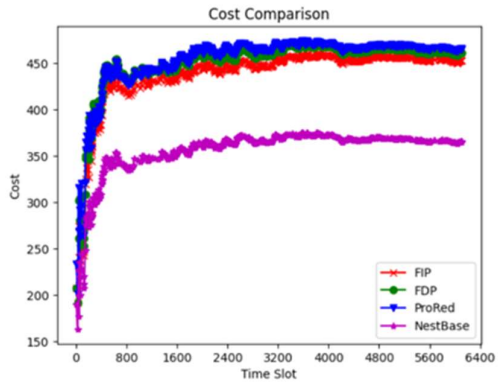
Fig.118 Average mapping (embedding) cost over time in Large SN with Ring VNs

(a)-(e): SN Link Probability 0.3-0.7

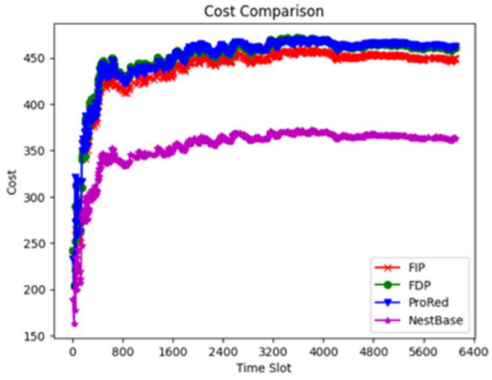
doi:10.6342/NTU202501591

Ring	0.3	0.4	0.5	0.6	0.7
FIP [16]	371	349	337	330	322
FDP [16]	381	361	349	342	332
ProRed [17]	348	349	353	352	354
NestBase	263	263	263	263	263

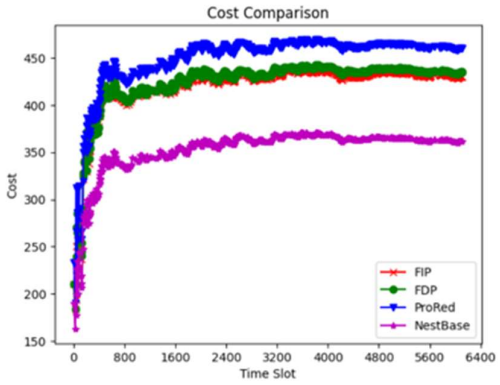
Table 31 Average mapping (embedding) cost in Large SN with Ring VNs: SN Link Probability 0.3-0.7



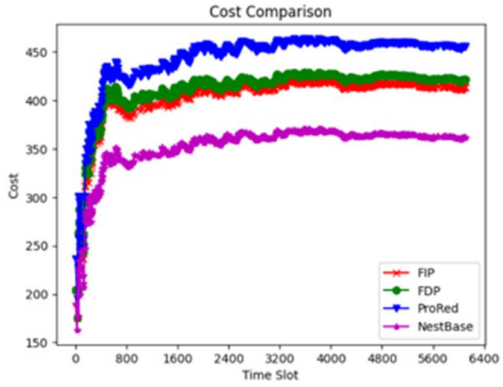
(a)



(b)

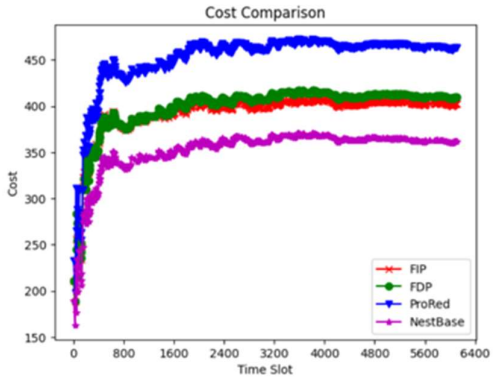


(c)



(d)

doi:10.6342/NTU202501591



(e)

Fig.119 Average mapping (embedding) cost over time in Large SN with Random VNs (a)-(e): SN Link Probability 0.3-0.7

Random	0.3	0.4	0.5	0.6	0.7
FIP [16]	452	449	430	414	402
FDP [16]	461	461	435	422	410
ProRed [17]	465	462	460	456	463
NestBase	365	363	362	362	362

Table 32 Average mapping (embedding) cost in Large SN with Random VNs: SN Link Probability 0.3-0.7

One key observation is that when the VN topology is simple (e.g., Line VN, Ring VN), the mapping costs in large SNs do not differ significantly from those in small SNs. This suggests that for these VN topologies, the relative performance among the three methods is not influenced by the scale of the SN (i.e., the number of nodes) but is instead more closely related to the sparsity of the SN (i.e., the link probability). Since the mapping costs remain consistent across different SN sizes, it indicates that SN sparsity plays a more significant role in determining the effectiveness of each method than SN size itself when dealing with simple VN topologies.

doi:10.6342/NTU202501591

A notable difference arises in complex VN topologies (Random VN), where the mapping costs in large SNs differ more significantly from those in small SNs. This indicates that in small SNs, there is evident competition among different VNs for SN resources, particularly when dealing with complex VN topologies.

Furthermore, when the SN link probability in small SNs exceeds 0.5, the mapping cost stabilizes and matches exactly with the mapping cost observed across all link probabilities in large SNs. This indicates that even in complex VN topologies, once the SN link probability surpasses 0.5 in a small SN, or when using a larger SN with a lower link probability, our Nest-Base algorithm can fully utilize SN resources without forcing different VNs to compete for SN resources, thereby avoiding suboptimal mapping decisions.

5.2.3.2 Acceptance Rate Evaluation

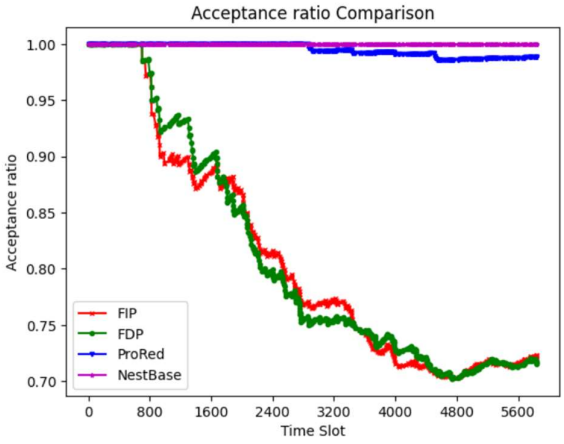


Fig.120 Comparison of VN acceptance ratio in Large SN with Line VNs

doi:10.6342/NTU202501591

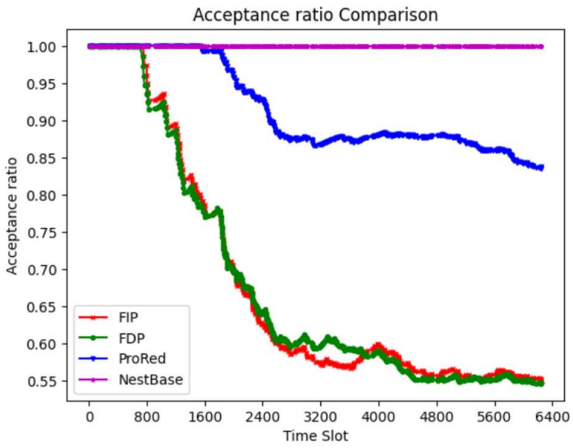


Fig.121 Comparison of VN acceptance ratio in Large SN with Ring VNs

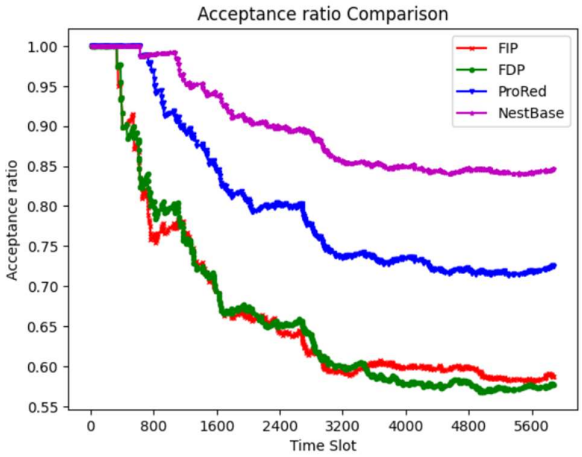


Fig.122 Comparison of VN acceptance ratio in Large SN with Random VNs

Under high-load conditions, [16] shows improved acceptance rates in large SNs compared to small SNs, reaching 0.7 in Line VN. However, in Ring VN and Random VN, the acceptance rate remains around 0.55, indicating that even with more SN resources, nearly half of the VNRs are still rejected. This suggests that [16] is unsuitable for high-load scenarios.

doi:10.6342/NTU202501591

ProRed achieves a high acceptance rate of 0.97 in Line VN, demonstrating its effectiveness in simple VN topologies. However, its performance drops by 12% in Ring VN and by 27% in Random VN, indicating a significant decline as VN complexity increases. This suggests that while ProRed performs well in simple VN topologies, it struggles to maintain high acceptance rates for more complex VNs under high-load conditions, making it unsuitable for handling diverse VN topologies in such scenarios.

Our Nest-Base algorithm achieves a perfect 100% acceptance rate in both Line VN and Ring VN, demonstrating its ability to fully utilize SN resources under high-load conditions. Even in Random VN, where acceptance rate decreases, it still maintains a strong 85% acceptance rate.

These results indicate that our method effectively manages SN resources, preventing VNR rejections even under extreme load conditions. Furthermore, even in complex VN topologies, Nest-Base consistently achieves an 85% acceptance rate over time, proving its robustness in handling diverse VNR demands in high-load scenarios.

doi:10.6342/NTU202501591

CHAPTER 6
CONCLUSIONS

This study addresses the Survivable Virtual Network Embedding (SVNE) problem under single facility node failures by introducing a decoupled approach that first augments the Virtual Network (VN) to ensure survivability before mapping (embedding) it onto the Substrate Network (SN).

Through our research, we discovered the relationship between the increment of backup resources in the augmented VN and the resulting mapping (embedding) cost. To fully exploit this relationship, we designed a mapping (embedding) algorithm that establishes a positive linear correlation between backup resource increments and mapping cost. This linearity ensures that optimizing the resource increment during the augmentation phase leads to a predictable and efficient reduction in mapping costs during the mapping (embedding) phase. By achieving this property, the augmentation process can be optimized in advance, reducing computational complexity while ensuring that the final mapping cost can be minimal.

Beyond the relationship between backup resource increments and mapping cost, we also identified the size of the augmentation strategy's solution space significantly affects how low the minimum mapping cost can be. Our analysis of various augmentation strategies led to the introduction of MUPL, an augmentation strategy that provides the largest solution space compared to existing methods. By ensuring that this solution space is sufficiently large, we confirmed that, in small-scale VNs, MUPL has the highest probability of containing the globally minimum mapping cost

doi:10.6342/NTU202501591

solution, which significantly improves the quality of the final embedding outcome.

To efficiently identify the optimal augmented VN within MUPL, we developed the Nest-Base algorithm, which rapidly selects the augmented VN with the minimal resource increment while maintaining computational feasibility. Combined with our proposed mapping algorithm, this algorithm ensures that the minimum mapping cost can be achieved during the embedding phase.

To validate our proposed methods, we conducted two types of evaluations. First, through exhaustive numerical analysis on small-scale VNs, we verified the linearity of our mapping algorithm and demonstrated that our Nest-Base algorithm consistently finds the globally optimal augmented VN within the MUPL solution space—one that leads to the minimum mapping cost, outperforming existing methods. Second, in realistic simulation scenarios across various SN and VN topologies, we observed that our approach reduced mapping costs by between 10% and 31.5% under moderate-load conditions compared to existing methods. Moreover, under high-load conditions, our method improved VNR acceptance rates by 10% to 45%, confirming its robustness in resource-constrained environments.

While this study successfully addresses the SVNE problem under single facility node failures, applying the decoupled approach to multiple simultaneous node failures remains a challenge. As the number of failure scenarios grows exponentially, the increased flexibility required in backup resource allocation further complicates the problem. However, since this decoupled approach effectively reduces complexity and efficiently finds mapping solutions, its extension to this problem remains a promising direction for future research.

doi:10.6342/NTU202501591

REFERENCES

[1] Chowdhury, N. M. K., & Boutaba, R. (2009). Network virtualization: state of the art and research challenges. IEEE Communications magazine, 47(7), 20-26.

[2] Chowdhury, N. M. K., Rahman, M. R., & Boutaba, R. (2009, April). Virtual network embedding with coordinated node and link mapping. In IEEE INFOCOM 2009 (pp. 783-791). IEEE.

[3] Fischer, A., Botero, J. F., Beck, M. T., De Meer, H., & Hesselbach, X. (2013). Virtual network embedding: A survey. IEEE Communications Surveys & Tutorials, 15(4), 1888-1906.

[4] Bhardwaj, S., Jain, L., & Jain, S. (2010). Cloud computing: A study of infrastructure as a service (IAAS). International Journal of engineering and information Technology, 2(1), 60-63.

[5] Rahman, M. R., Aib, I., & Boutaba, R. (2010). Survivable virtual network embedding. In NETWORKING 2010: 9th International IFIP TC 6 Networking Conference, Chennai, India, May 11-15, 2010. Proceedings 9 (pp. 40-52). Springer Berlin Heidelberg.

doi:10.6342/NTU202501591

[6] Yeow, W. L., Westphal, C., & Kozat, U. (2010, September). Designing and embedding reliable virtual infrastructures. In Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures (pp. 33-40).

[7] Haider, A., Potter, R., & Nakao, A. (2009, May). Challenges in resource allocation in network virtualization. In 20th ITC specialist seminar (Vol. 18, No. 2009). ITC.

[8] Shahriar, N., Ahmed, R., Chowdhury, S. R., Khan, M. M. A., Boutaba, R., Mitra, J., & Zeng, F. (2016, May). Connectivity-aware virtual network embedding. In 2016 IFIP Networking Conference (IFIP Networking) and Workshops (pp. 46-54). IEEE.

[9] Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C. N., & Diot, C. (2004, March). Characterization of failures in an IP backbone. In IEEE INFOCOM 2004 (Vol. 4, pp. 2307-2317). IEEE.

[10] Shahriar, N., Ahmed, R., Chowdhury, S. R., Khan, A., Boutaba, R., & Mitra, J. (2017). Generalized recovery from node failure in virtual network embedding. IEEE Transactions on Network and Service Management, 14(2), 261-274.

[11] Yu, H., Qiao, C., Anand, V., Liu, X., Di, H., & Sun, G. (2010, December). Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures. In 2010 IEEE Global Telecommunications Conference GLOBECOM 2010 (pp. 1-6). IEEE.

doi:10.6342/NTU202501591

[12] Chowdhury, S. R., Ahmed, R., Khan, M. M. A., Shahriar, N., Boutaba, R., Mitra, J., & Zeng, F. (2016). Dedicated protection for survivable virtual network embedding. IEEE Transactions on Network and Service Management, 13(4), 913-926.

[13] Yu, H., Anand, V., Qiao, C., & Sun, G. (2011, June). Cost efficient design of survivable virtual infrastructure to recover from facility node failures. In 2011 IEEE international conference on communications (ICC) (pp. 1-6). IEEE.

[14] Yuan, Y., Wang, C., Wang, C., Zhang, C., & Zhu, N. (2013, September). Fault tolerant virtual network embedding algorithm based on redundant backup resource. In 2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control (pp. 354-357). IEEE.

[15] Hu, Q., Wang, Y., & Cao, X. (2013). Survivable network virtualization for single facility node failure: A network flow perspective. Optical Switching and Networking, 10(4), 406-415.

[16] Guo, B., Qiao, C., Wang, J., Yu, H., Zuo, Y., Li, J., ... & He, Y. (2013). Survivable virtual network design and embedding to survive a facility node failure. Journal of Lightwave Technology, 32(3), 483-493.

[17] Ayoubi, S., Chen, Y., & Assi, C. (2016). Towards promoting backup-sharing in survivable virtual network design. IEEE/ACM Transactions on Networking, 24(5), 3218-3231.

doi:10.6342/NTU202501591

[18] Fajjari, I., Aitsaadi, N., Pujolle, G., & Zimmermann, H. (2011, December). Vnr algorithm: A greedy approach for virtual networks reconfigurations. In 2011 IEEE Global Telecommunications Conference-GLOBECOM 2011 (pp. 1-6). IEEE.

[19] Kuo-Liang Chang Chien, “Virtual Network Embedding in Faulty Fog Networks”, Graduate Institute of Communication Engineering College of Electrical Engineering and Computer Science National Taiwan University Master Thesis, 2019

[20] Chowdhury, M., Rahman, M. R., & Boutaba, R. (2011). Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. IEEE/ACM Transactions on networking, 20(1), 206-219.

[21] Cheng, X., Su, S., Zhang, Z., Wang, H., Yang, F., Luo, Y., & Wang, J. (2011). Virtual network embedding through topology-aware node ranking. ACM SIGCOMM Computer Communication Review, 41(2), 38-47.

[22] Yu-Hsiang Chao, “Virtual Network Embedding in Heterogeneous Fog Networks”, Graduate Institute of Communication Engineering College of Electrical Engineering and Computer Science National Taiwan University Master Thesis, 2020

[23] Wei-Che Chen, “Latency-Aware Virtual Network Embedding in Fog-Cloud Networks”, Graduate Institute of Communication Engineering College of Electrical Engineering and Computer Science National Taiwan University Master Thesis, 2019

doi:10.6342/NTU202501591

[24] Chowdhury, M., Samuel, F., & Boutaba, R. (2010, September). Polyvine: policy-based virtual network embedding across multiple domains. In Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures (pp. 49-56).

[25] Rahman, M. R., & Boutaba, R. (2013). SVNE: Survivable virtual network embedding algorithms for network virtualization. IEEE Transactions on Network and Service Management, 10(2), 105-118.

[26] Yu, H., Anand, V., Qiao, C., & Di, H. (2011, March). Migration based protection for virtual infrastructure survivability for link failure. In Optical Fiber Communication Conference (p. OTuR2). Optica Publishing Group.

[27] Khan, M. M. A., Shahriar, N., Ahmed, R., & Boutaba, R. (2015, November). Simple: Survivability in multi-path link embedding. In 2015 11th International Conference on Network and Service Management (CNSM) (pp. 210-218). IEEE.

[28] Yan-Jhu Wang, “Survivable Virtual Network Embedding Based on Single-Link Failures”, Graduate Institute of Communication Engineering College of Electrical Engineering and Computer Science National Taiwan University Master Thesis, 2023

[29] Yuan, Y., Wang, C. R., Wan, C., Wang, C., & Song, X. (2013). Repeatable optimization algorithm based discrete PSO for virtual network embedding. In Advances in Neural Networks–ISNN 2013: 10th International Symposium on Neural Networks, Dalian, China, July 4-6, 2013, Proceedings, Part I 10 (pp. 334-342). Springer Berlin Heidelberg.

doi:10.6342/NTU202501591