Towards Adaptive Joint Node and Link Mapping Algorithms for Embedding Virtual Networks: A Conciliation Strategy

Khoa Nguyen, and Changcheng Huang, *Senior Member, IEEE,*

***Abstract*—Network virtualization (NV) has emerged as a momentous facilitator for a notable triumph of future networks by allowing a flexibility, cost-efficiency and on-demand services through the deployment of heterogeneous network service requests on a shared physical infrastructure. The most major challenge of NV is to efficiently and effectively map diversified virtual network requests (VNRs), comprising a set of virtual nodes connected by virtual links, onto a shared substrate network meeting various stringent resource constraints. Most of the research papers in this field have merely focused on separate virtual node mapping (VNoM) or virtual link mapping (VLiM) with scalable heuristic algorithms for simple implementations. Unfortunately, the lack of a coordination between node and link mapping stages might cause low embedding results. In this paper, we present a new approach relied upon Genetic Algorithm (GA), that jointly coordinates virtual node and link mappings where the link mapping is based on three different path searching methods. Moreover, a novel heuristic conciliation mechanism is proposed to deal with a set of possibly infeasible link mappings while exploring embedding solutions within the operations of GA algorithm. Extensive performance results indicate that our proposed GA-based algorithms outperform state-of-the-art virtual mapping algorithms in all evaluation metrics we adopt.**

***Index Terms*—Network virtualization, joint node-link mapping, conciliation strategy, Genetic Algorithm, virtual network embedding.**

I. Introduction

NETWORK virtualization (NV) has envisioned as a de facto paradigm for the foreseen triumph of future networks such as virtualized fifth generation [1], and smart Internet of Things (IoT) networks [2]. The adoption of NV enables to share physical resources amongst multiple virtual network requests (VNRs), providing an isolated coexistence of a number of virtual network (VNs) on a shared underlying substrate network (SN). Virtualization efficiently enhances physical network utilization, and simplifies the development and evaluation of new network protocols or architecture designs without unexpected expansion.

The fundamental concept of NV is to decouple a typical role of an Internet Service Provider (ISP) into two separate business components as illustrated in Fig. 1: Infrastructure Providers (InPs) and Service Providers (SPs). InPs will manage the substrate resources, and control Quality of Service (QoS) at the network layer while SPs will aggregate network resources that are leased from different InPs into VNs providing customised services to end users. In a virtualization environment, a network service provider (SP) typically converts an application or service into a VN which is then transferred to a corresponding infrastructure provider (InP) as a form of a VNR as illustrated in Fig. 2. The InP attempts to map such VN onto its substrate infrastructure meeting several rigorous resource constraints by adopting an optimization process. In terms of InPs, they attempt to approach an efficient resource allocation algorithm that is capable to maximize their generated revenues by accepting as many VNRs as possible while keeping the mapping costs minimal.

A conventional VNR is consisted of a set of virtual nodes that are connected by virtual links in order to establish an explicit topology. In most of scenarios, it dynamically arrives and resides in the corresponding network for an arbitrary duration. Mapping VNRs onto the shared physical infrastructure with varied topology and stringent resource demands is commonly recognized as a virtual network embedding (VNE) problem. VNE is widely known as a generalization of second stage of

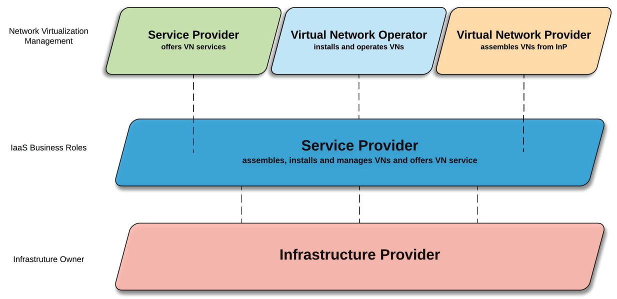


Fig. 1: *Towards the Future internet Model [3]*

network function virtualization resource allocation (NFV-RA), namely virtual network function forwarding graph embedding (VNF-FGE) since VNE is in the same problem domain with NFV-RA as the goal of both problems is to efficiently allocate VNRs on the top of SN infrastructure [4]. VNF-FGE is also the central problem of NFV-RA accompanying with virtual network functions (VNFs) chain composition (known as service function chaining) and VNFs scheduling, where the chain composition stage has been usually neglected [4]. Additionally, VNE is assumed to be more complicated than VNF-FGE in some specific cases and topology [5], [6].

VNE is validated to be $\mathcal{NP}$-Hard either towards VNoM or VLiM [4], [7]. Integer Linear Programming (ILP) is usually

Khoa Nguyen and Changcheng Huang are with the Department of Systems and Computer Engineering, Carleton Univeristy, Ottawa, ON, CA e-mail: {khoatnguyen, huang}@sce.carleton.ca. Some primary results of GA-SEQ algorithm in this paper are partly presented at the 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), 2021, online. This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Engage grant (RGPIN-2017-06331).

Corresponding author: Khoa Nguyen (e-mail: khoatnguyen@sce.carleton.ca)

formulated to solve the VNE problem, but it encounters many challenges since its model has to confront with scalability, intricate deployment as well as time complexity. In fact, Exact solutions are not apt to deal with the online VNE problem. The majority of VNE studies have concentrated on efficient designs of heuristics or metaheuristics to tackle the aforementioned obstacles of the Exact optimization models. Similar to VNF-

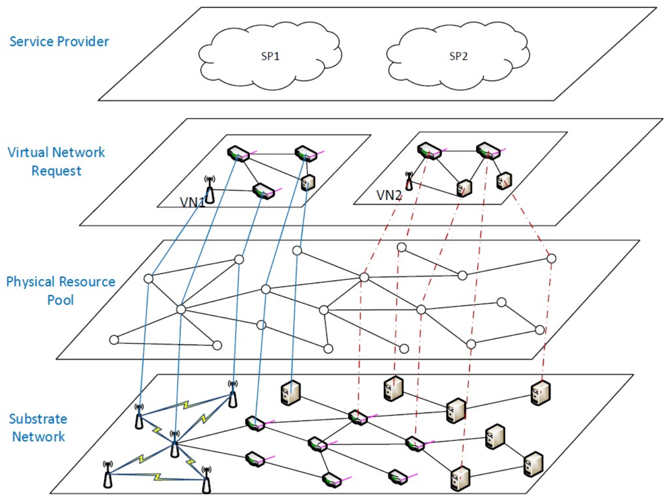


Fig. 2: *The concept of network virtualization [8]*

FGE, VNE process consists of two stages: VNoM and VLiM. Up to now, most VNE approaches have been proposed to uncoordinately solve VNoM phase following VLiM phase. There are a large number of solutions for VNoM problem whereas VLiM is usually relied on shortest path methods (e.g., Dijkstra's algorithm), multi-commodity flow (MCF) and recently distributed parallel GA algorithms [9]. The decoupling can facilitate the complexity of algorithmic deployments, but this common approach is most likely to sacrifice some degrees of optimality. Lack of coordination between VNoM and VLim stages would lead to low acceptance ratio and correspondingly low network revenue as well as high cost. For instance, although VNoM stage can figure out highly promising substrate nodes for mapping virtual nodes in a VNR, we may fall into an unexpected situation that there are no paths in the SN that have sufficient remaining resources to connect one or more pairs of those virtual nodes together with respect to node mappings in previous phase. Then, the corresponding VNR is obviously rejected. In practice, the most common failures of embedding VNRs onto the shared underlying SN are usually derived from the infeasible link mappings [10]. Indeed, those rejected VNRs can be reassessed after the link mapping stage or be put back into the queue and waiting until the network is idle. These approaches are inefficient and unpractical since they could violate delay constraints of VNRs, and obviously waste the network resources for processing such failed VNRs in the past. As a result, it is required VNE algorithms that simultaneously consider node and link mappings, and allow an efficient remapping approach if these node mappings are inappropriate due to the failure of link mappings. Designing VNE algorithms meeting these demands is still in its infancy.

In this paper, we propose metaheuristic-based algorithms to solve VNE problem by jointly mapping virtual nodes and corresponding virtual links. GA algorithm attempts to randomly find potential solutions for all virtual nodes in a VNR. Then, various path searching methods are conducted to seek for the corresponding virtual links sequentially. If no feasible paths for some possible link requests are achieved due to network congestion for example, corresponding mappings should be reassessed. Because the node mappings influence the link mappings accordingly, we should figure out an intelligent mechanism that considers the minimal number of virtual nodes and virtual links for reassessments. Thus, we introduce an efficient conciliation algorithm to address this problem. Through iterative operations, GA algorithm, driven by an effective fitness function that considers both network resources and virtual resource demands, generates the population of several promising VNE solutions including node and link mappings. Furthermore, we present a distributed parallel GA-based operation scheme employing a set of distributed parallel machines to reduce the operation time. A brief comparison towards execution time between sequential and parallel manners is also provided. Splittable mapping may theoretically achieve better resource utilization, but this solution could generate abundant overhead in order to maintain consistent network state and likely results in out-of-order delivery problem with extra latency that could be unacceptable for sensitive-delay applications [11]. Thus, we only consider unsplittable-enabled mapping solutions in this paper.

Major contributions of this paper are summarized as below:

- We propose a new joint node and link mapping approach based on GA algorithm to efficiently and effectively embed heterogeneous VNRs, where the link mapping is relied on three different virtual link mapping algorithms including random path searching, sequential single path searching and *k*-shortest path method. We manage node and link mappings in a coordinated manner so when node mappings are changed, the link mappings are altered accordingly.
- To deal with a possible set of infeasible link mappings caused by improper node mappings, we introduce a novel heuristic conciliation mechanism which is aimed at keeping the number of both remapped nodes and links minimal.
- An efficient fitness function with various network resource factors including embedding cost, hop-count, and propagation delay is introduced to drive GA algorithm to near-optimal VNE solutions.
- We advise a distributed parallel operation scheme to decrease time complexity of GA algorithm. We also implement our proposed GA-based algorithms (according with three different link mappings) on both sequential and parallel manners, and then compare execution time between them.

The rest of this paper is organized as follows. Section II presents the related work while Section III formulates the network model. Jointly node-link VNE embedding is described in Section V with three variances. Performance evaluation is then introduced in Section VI. Section VII finally presents our conclusions and future work.

II. Related Work

[3], [12], [13] and [14] have contributed comprehensive surveys in network virtualization field. The VNE problem, widely known as $\mathcal{NP}$-hard in nature, is intractably solved by Integer Programming (IP). Thus, a large number of research papers have focused on heuristic algorithms due to time complexity of Exact methods. Chowdhury et al. in [7] introduced a VNE coordination between node and link mapping phases for embedding virtual nodes with respect to minimizing the embedding cost. A Mixed Integer Programming (MIP) was formulated to solve VNoM considering meta nodes on an augmented graph created over a physical network. Integer constraints of the MIP model were relaxed to reduce its time complexity. Authors in [15] proposed VNE Node-Link Formulation (VNE-NLF), an ILP model, for mapping each VNR. Its objective was to minimize the VN embedding cost while supporting load balancing. Huang et al. in [16] extended [7] by taking both node splitting and node collocation into account for solving VNE efficiently. The authors studied a realistic VNE scenario in which virtual and substrate nodes were managed as transit nodes or stub nodes that were accountable for traffic reply or traffic originator/sinker. Authors in [17] presented a topology-aware node mapping based on the Markov Random Walk model to evaluate network resources. A node-ranking approach relied on node degree and node clustering coefficient information to construct the metric of node importance for ranking the underlying physical nodes was proposed in [18]. Moreover, the authors in [19] and [20] studied the attributes of network topology and the global network resources in order to rank substrate nodes for VNoM stage. To improve the possibility of being selected of a substrate node, its physical resources and the relationship with other nodes in the network were considered. These node-ranking approaches handled network resource fragmentation efficiently. Authors in [8] tackled the coordination of VNE with the Mod-MaxMatch algorithm for VNoM aimed at minimizing the substrate links reused, and then the path-splitting enabled GA algorithm was proposed for VLiM.

GA algorithm has been widely applied in many optimization fields including VNE. Its applications on VNE were first investigated in [21]. The authors proposed GA-based node ranking methods relied on multiple topology attributes and residual network resources. Zhou et al. in [22] advised GA algorithm for embedding multiple VNRs simultaneously under batched arrivals, and designed a resource-aware mechanism for maximizing long-term revenue. Zhang et al. modified a conventional GA algorithm by rearranging the mutation operator before the selection to efficiently explore the search space in [23]. Meanwhile, a hybrid solution, that combined an adaptive GA algorithm with a new crossover operator to enhance GA's convergence and simulated annealing algorithm replacing a traditional mutation operation of GA to avoid premature VNE solutions, was introduced in [24]. The authors in [25] and [26] applied GA algorithm to address VNE problem in multiple InP domains. The solution allowed to embed multiple VNRs on different InPs owning several SNs. However, it faced the problems of scalability and complex implementations. On the other hand, our previous work in [9], [27] presented a novel GA-based approach for VLiM stage, which validated the important role of VLiM in VNE problems. Although these GA-based approaches performed better than other metaheuristics and several heuristic algorithms, they solved VNE problems uncoordinatedly, leading low embedding efficiency [3].

Recently, several papers have studied the applicability of Reinforcement Learning (RL) techniques on VNE problems. Haeri and Trajkovic [28] modeled VNE problem as Markov Decision Process (MDP) for solving VNoM stage and determined action policies using Monte Carlo Tree Search (MCTS) that was driven by a reward function. Yao et al. [29] presented a RL-based model utilizing a dynamic attribute matrix representation (RDAM) to address VNE problem. As such, the attributes of network topology as well as its specific structures under a matrix representation were considered. Another extension of [29] was presented in [30] that was inspired by the node ranking method. To train agents, three network attributes were extracted for each physical node to construct a feature matrix that was utilized as the input of the policy network in a Convolutional Neural Network (CNN). Simple BFS mechanism was then deployed in VLiM stage. Extending [30] to five network features with a security-aware focus was presented in [31]. In addition, Recurrent Neural Networks (RNNs) were used in [32] in order to model the continuity of VNoM process in time series formulated as a classic seq2seq, aimed at exploiting the historical states of SN. A continuous-decision VNE scheme based on RL relied on policy-gradient mechanism updated the RNN parameters with respect to average long-term revenue-to-cost ratio. However, the aforementioned VNE approaches merely carry on VNs mapping in separate stages, including our previous work in [9], [27] where VLiM was the focal center of interest. This paper is aimed at coordinately joining node and link mappings in a single stage based on GA algorithm. Different path searching methods including random path searching, sequential path searching mechanisms and shortest path method for mapping virtual links are proposed. To handle failed virtual link mappings due to inappropriate node mappings, a novel conciliation mechanism is also introduced.

Taxonomy of the VNE approaches is shown in Table I where the category classifies each approach into four features **O**nline or o**F**fline, **C**entralized or **D**istributed, **S**tatic or **D**ynamic and **C**oncise or **R**edundant, whereas the coordination is involved in **CO**or**D**inated or **U**n**C**oor**D**inated approaches [3].

III. Network Model and Problem Descriptions

A. Virtual Network Assignment

We model the SN as a weighted undirected graph $G^s = (N^s, L^s)$, in which N^s and L^s are associated with the sets of substrate nodes and substrate links, respectively. Each physical node $m^s \in N^s$ has its geographical location $loc(m^s)$ and available CPU capacity $C(m^s)$, whereas a substrate link $(m^s, n^s) \in L^s$ between two substrate nodes occupies a $B(m^s n^s)$ bandwidth capacity. Similarly, the i^{th} VNR arriving in the network is modelled as a weighted undirected graph denoted as $G_i^v = (N_i^v, L_i^v)$, in which N_i^v is the set of virtual nodes and L_i^v is the set of virtual links. Each virtual node

TABLE I: Taxonomy of VNE approaches

Category	Reference	Optimization	Coordination	Contribution
O/C/S/R	[17] Cheng *et al.* (2011)	Heuristic	COD, two-stage	The topology-aware node ranking approach based on Markov Random Walk model in VNE.
O/C/S/R	[7] Chowdhury *et al.* (2012)	Like-exact	COD, two-stage	Relaxed MIP model considering a coordinated node and link approach for VNoM using k-shortest path method or MCF for VLiM.
O/C/S/C	[21] Mi *et al.* (2012)	Metaheuristic	UCD, two-stage	One of the first GA-based VNE algorithms for VNoM.
O/C/S/R	[15] Melo *et al.* (2013)	Exact	COD, one-stage	VNE Node-Link Formulation (VNE-NLF) based on the ILP model for mapping VNRs.
O/C/S/C	[20] Feng *et al.* (2014)	Heuristic	COD, two-stage	Seven topological SN characteristics complementary to each other for node ranking in VNE.
O/C/S/C	[18] Zhang *et al.* (2016)	Heuristic	UCD, two-stage	A VNE node ranking approach based on the node degree and clustering coefficient information.
O/C/S/R	[22] Zhou *et al.* (2016)	Metaheuristic	COD, two-stage	GA-based approach for embedding multiple VNRs under batched arrivals in VNE.
O/C/S/C	[25] Pathak *et al.* (2017)	Metaheuristic	UCD, two-stage	GA-based algorithm for mapping multiple VNRs amongst multiple InPs.
O/C/S/R	[16] Huang *et al.* (2018)	Exact	COD, two-stage	Node splitting and node collocation for VNoM.
O/C/S/C	[19] Cao *et al.* (2018)	Heuristic	COD, two-stage	Node-ranking approach considering five network topology attributes and global network resources.
O/C/S/R	[28] Haeri *et al.* (2018)	RL	UCD, two-stage	MDP-based RL using the Monte Carlo tree search algorithm for VNoM.
O/C/S/C	[29] Yao *et al.* (2018)	RL	UCD, two-stage	RL based dynamic attribute matrix representation algorithm for VNoM.
O/C/S/C	[23] Zhang *et al.* (2019)	Metaheuristic	UCD, two-stage	A modified GA-based approach rearranging mutation operator right after initial population to explore the searching space efficiently.
O/C/S/C	[24] Boyang *et al.* (2019)	Metaheuristic	UCD, two-stage	A hybrid adaptive GA-based solution combining an adaptive GA algorithm with new crossover and simulated annealing to improve convergence, and avoid premature phenomenon.
F/C/S/R	[8] Huang *et al.* (2019)	Heuristic	COD, two-stage	Mod-MaxMatch algorithm for VNoM and the path-splitting enabled GA-based algorithm for VLiM.
O/C/S/C	[9] Nguyen *et al.* (2020)	Metaheuristic	UCD, two-stage	Distributed parallel GA-based algorithm for VLiM with a novel elastic crossover operation and a multi-objective fitness function.
O/C/S/C	[27] Lu *et al.* (2020)	Metaheuristic	UCD, two-stage	Two distributed parallel GA-based algorithms based on two versions of crossover and mutation schemes for VLiM.
O/C/S/C	[30] Zhang *et al.* (2020)	RL	UCD, two-stage	Constructing an agent learning environment based on the extracted node features of the SN for training the RL model in CNN.
O/C/S/C	[31] Zhang *et al.* (2020)	RL	UCD, two-stage	A security-aware VNE algorithm based on RL.
O/C/S/C	[32] Yao *et al.* (2020)	RL	UCD, two-stage	A continuous-decision RL-based VNE scheme updating RNN's parameters by the policy-gradient mechanism.
O/C/S/C	[26] Zhang *et al.* (2021)	Metaheuristic	UCD, two-stage	A genetic correlation multi-domain VNE algorithm with a stochastic selection and a feasibility checking.
O/C/S/C	This paper	Metaheuristic	COD, one-stage	Joint GA-based node and link mapping approaches, where VLiM is based on several link mapping algorithms. A novel conciliation strategy cleverly handles a set of infeasible link mappings.

$u^v \in N_i^v$ has a CPU demand $C(u^v)$, while a virtual link $(u^v, k^v) \in L_i^v$ from a virtual source node u^v to a virtual destination node k^v possesses a required bandwidth demand $B(u^v k^v)$. Each VNR generally prefers an embedding radius $D(u^v)$ revealing the farthest distance the virtual node u^v is allowed to map from its location $loc(u^v)$. $D(u^v)$ can be expressed in reference to physical distance or concerning permissible delay (e.g., propagation delay) from $loc(u^v)$ similar to [7]. Without loss of generality, memory and storage are ignored for simplification.

Our objectives are to maximize average acceptance ratio, average revenue over cost ratio, and to improve node and link utilization and propagation delay, meeting node and link constraints imposed by VNRs.

Node constraints:

$$C(u^v) \leq R_N(\mathcal{A}_\mathcal{N}(u^v)) \tag{1}$$

$$\mathcal{D}(loc(u^v), loc(\mathcal{A}_\mathcal{N}(u^v))) \leq D(u^v) \tag{2}$$

$$\mathcal{A}_\mathcal{N}(u^v) \in N^s \tag{3}$$

$$R_N(m^s) = C(m^s) - \sum_{u^v \to m^s} C(u^v) \tag{4}$$

where $\mathcal{A}_{\mathcal{N}}(u^v)$ is the mapping solution of virtual node u^v. $\mathcal{D}(i^s, j^d)$ and $R_N(m^s)$ denote the distance between i^s and j^d, and the residual CPU capacity of a physical node respectively. Eq. (1), (2) ensure that the residual CPU capacity and mapping distance of the substrate node meet the requested CPU demand and preferred embedding radius of the corresponding virtual node, respectively. Eq. (3) guarantees that the mapped substrate node belongs to the physical network, whereas Eq. (4) is the measurement of available CPU capacity of the substrate node m^s.

Link constraints:

$$R_L(e^s) \geq B(u^v k^v), \forall e^s \in \mathcal{E}^s(\mathcal{A}_{\mathcal{L}}(u^v k^v)) \tag{5}$$

$$R_L(e^s) = \min_{m^s n^s \in e^s} R_L(m^s n^s) \tag{6}$$

$$R_L(m^s n^s) = B(m^s n^s) - \sum_{u^v k^v \to m^s n^s} B(u^v k^v) \tag{7}$$

where $\mathcal{E}^s(\mathcal{A}_{\mathcal{L}}(u^v k^v))$ is a set of all possible substrate paths from the source node $\mathcal{A}_{\mathcal{N}}(u^v)$ to the destination node $\mathcal{A}_{\mathcal{N}}(k^v)$. $R_L(e^s)$ denotes the available bandwidth of a substrate path $e^s \in \mathcal{E}^s$, while $R_L(m^s n^s)$ represents the remaining capacity of a physical link. Similarly, Eq. (5) ensures the remaining bandwidth capacity of the substrate path meeting the requested bandwidth request of the virtual link, whereas Eq. (6) and (7) are available bandwidth capacity of the substrate path e^s and the measurement of residual capacity of the substrate link. A mapping solution is called "feasible" if and only if it meets resource constraints (1)-(4) of node mappings and (5)-(7) of link mappings. A path for a mapped virtual link in the feasible solution is therefore called a feasible path.

B. Performance metrics

The main objectives of VNE problems are associated with maximizing the generated revenues of InPs while minimizing the corresponding mapping cost. In this work, an InP's revenue $\Gamma(G_i^v)$ is defined as the sum of all virtual resources successfully embedded on the underlying SN over time, whereas the mapping cost of the i^{th} VNR, denoted as $\Xi(G_i^v)$, is the total network resources required for allocating the i^{th} virtual network [7].

Revenue of i^{th} VNR G_i^v is formulated as follows:

$$\Gamma(G_i^v) = w_b * \sum_{u^v k^v \in L_i^v} B(u^v k^v) + w_n * \sum_{u^v \in N_i^v} C(u^v) \tag{8}$$

where $B(u^v k^v)$ is the bandwidth demand of virtual link $u^v k^v$ and $C(u^v)$ is the CPU capacity request of virtual node u^v. Besides, w_b and w_n denote the unit weights preferable to bandwidth or CPU resources, respectively.

Cost of i^{th} VNR G_i^v can be formulated as below:

$$\Xi(G_i^v) = \sum_{u^v \in N_i^v} C(u^v) + \sum_{u^v k^v \in L_i^v} \sum_{m^s n^s \in L^s} f_{m^s n^s}^{u^v k^v} \tag{9}$$

where $f_{m^s n^s}^{u^v k^v}$ denotes bandwidth capacity of the physical link (m^s, n^s) allocated to the virtual link (u^v, k^v).

The average revenue over cost ratio can be indeed utilized to estimate how efficient a VNE algorithm is. High average acceptance ratio as well as high average revenue to cost ratio are highly desired because the results can disclose that substrate resources are efficiently allocated.

Revenue to cost ratio (R/C) of i^{th} VNR G_i^v is formulated as follows:

$$\Upsilon(G_i^v) = \frac{\Gamma(G_i^v)}{\Xi(G_i^v)} = \frac{w_b * \sum_{u^v k^v \in L_i^v} B(u^v k^v) + w_n * \sum_{u^v \in N_i^v} C(u^v)}{\sum_{u^v \in N_i^v} C(u^v) + \sum_{u^v k^v \in L_i^v} \sum_{m^s n^s \in L^s} f_{m^s n^s}^{u^v k^v}} \tag{10}$$

Acceptance ratio: is the fraction between the number of accepted VNRs over the total number of VNRs arrived in the network during a time interval τ is computed as follows:

$$\mathcal{A}_c^\tau = \left| \frac{\xi^a(\tau)}{\xi(\tau)} \right| \tag{11}$$

where $\xi^a(\tau)$ and $\xi(\tau)$ denote the total number of successfully mapped VNRs and the total number of VNRs, respectively.

Node utilization indicates the distribution of network loads on the corresponding physical network. It is equal to the total network resources that are occupied by virtual node requests during a certain time divided by total resources of substrate nodes. Node utilization can be expressed as below:

$$\mathcal{U}_N(G^s) = \sum_{m^s \in N^s} \left(\frac{\sum_{u^v \to m^s} C(u^v)}{C(m^s)} \right) * T_i \quad , \tag{12}$$

where T_i is the time duration of the accepted i^{th} VNR.

Link utilization: Similarly, link utilization can be presented as follows:

$$\mathcal{U}_L(G^s) = \sum_{m^s n^s \in L^s} \left(\frac{\sum_{u^v k^v \to m^s n^s} B(u^v k^v)}{B(m^s n^s)} \right) * T_i \quad , \tag{13}$$

Fitness Function (FF) assesses the quality of VNE solutions that can be reproduced in next generations. Fitness values serve as rewards to help guide the searching process for the optimal solutions. In this paper, FF takes the mapping cost, hop-count and propagation delay of the mapping solutions into consideration and decides which is the best solution for a VNR. Due to the coordinated process of our proposed approach, these factors reflect total network resources including both CPU and bandwidth as well as network latency. Additionally, hop-count factor and propagation delay feature allow to select neighboring substrate nodes, which possibly reduces the resource fragmentation problem, and enhances the possibility of accepting future VNRs. Our fitness function $\mathcal{F}(\mathcal{S}_z)$ can be expressed as follows:

$$\mathcal{F}(\mathcal{S}) = \left(\frac{1}{\Xi(G_i^v)}\right) * w_c + \left(\frac{1}{\sum_{\forall (u^v, k^v) \in L_i^v} H_{\mathcal{A}_{\mathcal{L}}(u^v k^v)}}\right) * w_h + \left(\frac{1}{\sum_{\forall (u^v, k^v) \in L_i^v} D_{\mathbb{P}}(\mathcal{A}_{\mathcal{L}}(u^v k^v))}\right) * w_p \tag{14}$$

where, $\mathcal{S}$ is a feasible solution whereas H and $D_{\mathbb{P}}$ define hop-count factor and propagation delay of the link mapping of (u^v, k^v) respectively. w_c, w_h, and w_p denote the predefined

TABLE II: *List of Notations*

$G^s(N^s, L^s)$	Substrate network
$G_i^v(N_i^v, L_i^v)$	i-th Virtual network
m^s	Substrate node, $m^s \in N^s$
$m^s n^s$	Substrate link, $(m^s, n^s) \in L^s$
u^v	Virtual node, $u^v \in N_i^v$
$u^v k^v$	Virtual link, $(u^v, k^v) \in L_i^v$
$loc(m^s)$	Geographical location of n^s
$C(m')$	CPU capacity of m', m' can be m^s or u^v
$B(l')$	Bandwidth capacity of l', l' can be $u^v k^v$ or $m^s n^s$
$f_{m^s n^s}^{u^v k^v}$	bandwidth capacity of $m^s n^s$ allocated to $u^v k^v$
$\mathcal{D}(u^v, m^s)$	Distance between u^v and m^s
$D(u^v)$	Preferable mapping radius
$D(u^v k^v)$	Propagation delay or physical distance demand
$\Gamma(G_i^v)$	Generated revenue of G_i^v
$\Xi(G_i^v)$	Mapping cost of G_i^v
$\Upsilon(G_i^v)$	Revenue to cost ratio of G_i^v
$\mathcal{A}_c^\tau$	Acceptance ratio during an interval time τ
$\xi^a(\tau)$	Successfully embedded VNRs in time τ
$\xi(\tau)$	Number of VNRs arrived in time τ
$\mathcal{U}_N$	Node utilization of G^s
$\mathcal{U}_L$	Link utilization of G^s
$\mathcal{A}_\mathcal{N}(u^v)$	Mapping of the virtual node u^v
$\mathcal{A}_\mathcal{L}(u^v k^v)$	Mapping of the virtual link $u^v k^v$
$\mathcal{E}^s(\mathcal{A}_\mathcal{L}(u^v k^v))$	The set of all available paths of $\mathcal{A}_\mathcal{L}(u^v k^v)$
e^s	A substrate path, $e^s \in \mathcal{E}^s$
$R_N(m^s)$	Residual CPU capacity of m^s
$R_L(e')$	Residual bandwidth capacity of e', e' can be e^s or (m^s, n^s)
$\mathcal{F}(\mathcal{S})$	Fitness function of a feasible solution $\mathcal{S}$
H	Hop-count
$D_\mathbb{P}$	Propagation delay
$x_{m^s}^{u^v}$	Binary node variable
$y_{m^s n^s}^{u^v k^v}$	Binary link variable
$\mathcal{N}^s(m^s)$	A set of neighbors of a physical node $m^s \in N^s$
$\mathcal{F}_i^v$	A set of failed virtual links
$\mathcal{E}_i^r$	Remapping list
$Q, \mathcal{Q}, \mathcal{R}$	Trials in algorithms

weights of cost, hop-count and propagation delay factors, respectively. List of notations is shown in Table II.

IV. Mathematical Formulation

This section describes the mathematical formulation to solve the online VNE problem with several defined constraints. A virtual node must be embedded on a physical node while a virtual link must be mapped to a substrate path. The following decision variables indicate node and link mappings.

A. Variables

$$x_{m^s}^{u^v} = \begin{cases} 1, & \text{virtual node } u^v \in N_i^v \text{ is allocated to} \\ & \text{substrate node } m^s \in N^s \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

$$y_{m^s n^s}^{u^v k^v} = \begin{cases} 1, & \text{virtual link } (u^v, k^v) \in L_i^v \text{ is assigned} \\ & \text{to a substrate link } (m^s, n^s) \in L^s. \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

B. Objectives:

we propose an objective function as below:

$$\textit{Minimize} \quad \alpha_u \Big(\sum_{u^v \in N_i^v} C(u^v). \sum_{m^s \in N^s, u^v \in N_i^v} \frac{x_{m^s}^{u^v}}{R_N(m^s) + \delta} \Big) + \beta_{mn} \Big(\sum_{u^v k^v \in L_i^v} B(u^v k^v). \sum_{u^v k^v \in L_i^v, m^s n^s \in L^s} \frac{y_{m^s n^s}^{u^v k^v}}{R_L(m^s n^s) + \delta} \Big) \tag{17}$$

C. Constraints

$$\forall u^v \in N_i^v, \forall m^s \in N^s, \quad \sum_{u^v} x_{m^s}^{u^v}.C(u^v) \leq R_N(m^s) \tag{18}$$

$$\forall (m^s, n^s) \in L^s, m^s \in \mathcal{A}_\mathcal{N}(u^v), n^s \in \mathcal{A}_\mathcal{N}(k^v), m^s \neq n^s, \quad \sum_{\forall (u^v, k^v) \in L_i^v} y_{m^s n^s}^{u^v k^v}.B(u^v k^v) \leq R_L(m^s n^s) \tag{19}$$

$$\forall (m^s, n^s) \in L^s, m^s \in \mathcal{A}_\mathcal{N}(u^v), n^s \in \mathcal{A}_\mathcal{N}(k^v), m^s \neq n^s, \quad \sum_{\forall (u^v, k^v) \in L_i^v} y_{m^s n^s}^{u^v k^v}.D_\mathbb{P}(m^s n^s) \leq D(u^v k^v) \tag{20}$$

$$\forall u^v \in N_i^v : \sum_{\forall m^s \in N^s} x_{m^s}^{u^v} = 1 \tag{21}$$

$$\forall m^s \in N^s : \sum_{u^v \in N_i^v} x_{m^s}^{u^v} \leq 1 \tag{22}$$

$$\forall (u^v, k^v) \in L_i^v : \sum_{\forall (m^s, n^s) \in L^s} y_{m^s n^s}^{u^v k^v} \geq 1 \tag{23}$$

$$\forall (u^v k^v) \in L_i^v, \forall (m^s, n^s) \in L^s, \forall m^s \in N^s : \quad \sum_{\forall n^s \in \bar{\mathcal{N}}^s(m^s)} y_{m^s n^s}^{u^v k^v} - y_{n^s m^s}^{u^v k^v} = x_{m^s}^{u^v} - x_{m^s}^{k^v} \tag{24}$$

$$\forall u^v \in N^v, \forall m^s \in N^s, x_{m^s}^{u^v} \in \{0, 1\} \tag{25}$$

$$\forall (u^v, k^v) \in L^v, \forall (m^s, n^s) \in \mathcal{E}^s(\mathcal{A}_\mathcal{L}(u^v k^v)), \ y_{m^s n^s}^{u^v k^v} \in \{0, 1\} \tag{26}$$

Remarks:

- Function (17) is the objective function that tries to minimize the total cost while balancing the loads. By dividing the remaining resources, the network loads can be normalized and balanced. It is also aimed at minimizing the number of substrate links consumed to the minimum possible. Since the requested demands of the VNR are included in objective function, nodes and links with more residual resources that tend to generate less embedding costs are preferable. δ is a small positive number used to avoid dividing by zero. α_u and β_{mn} are weight factors to determine the significance of the corresponding loads of either links or nodes, $\alpha_u > 0$ and $\beta_{mn} > 0$.
- Constraints (18), (19) and (20) ensure that the available CPU, bandwidth capacity and link delay are not violated respectively.
- Constraints (21) and (22) guarantee that each virtual node is only assigned to a substrate node and vice versa. Constraint (23) ensures that each virtual link is assigned to a subset of substrate links or a path.

- Constraint (24) is considered as a flow-conservation, guaranteeing that virtual links are mapped to physical links, where $\bar{\mathcal{N}}^s(m^s)$ denotes a set of neighbors of a physical node $m^s \in N^s$.
- Finally, constraints (25) and (26) denote binary domain constraints on variables $x^{u^v}_{m^s}$ and $y^{u^v k^v}_{m^s n^s}$, respectively.

V. Joint node-link Mapping

A. Background of metaheuristic algorithms

Metaheuristics comprise a set of optimization techniques aimed at achieving the global optimum by discovering the search space effectively. Intrinsically, metaheuristics exploit diversified variation operations to explore new potentials systematically, and then their multi-objective fitness function will drive such achieved potentials to the optimum. Approaching an effective algorithm satisfying several stringent constraints becomes challenging. In fact, metaheuristic methods have been successfully employed in various applications from different fields, including operation research, industrial engineering to management science. Evolutionary Computation (EC) that imitates the natural evolution process consists of a set of metaheuristic algorithms for seeking the global optimization. Genetic Algorithm, a mature metaheuristic motivated by the Darwin evolution principle through natural selection, is one of the most popular population-based metaheuristic methods in EC. It is suitable for solving multi-objective linear or non-linear programming optimization problems due to its simplicity and ease of implementation. GA is fast and more efficient than many heuristic methods by maintaining a balance between exploration and exploitation [21]–[26].

Furthermore, GA-based approaches bear a resemblance to reinforcement learning (RL) algorithms in the aspect that they both interact with environments through an iterative action-reward process. However, to our best knowledge, modern RL algorithms are focused on integrating deep learning techniques. The environments are typically modelled as a Markov Decision Process with transition probabilities as parameters to be learned through deep learning techniques. Those parameters are assumed static so that the learning process can estimate them and used the model with these parameters to decide future actions. Our GA-based approach does not need to estimate those parameters. Fundamentally, upon the arrival of a VNR, it does a smart search to find the best result it can. The only assumption it uses is that children are likely better than their parents. This makes it adaptable to the dynamic situations where loads keep changing. Due to similarities, GA is also advised as a scalable alternative to the cutting-edge RL algorithms [33]. Even GA does not always outperform RL algorithms, but GA is considerably faster than RL since it exposed great scalability and parallel capabilities [33].

A typical GA algorithm comprises four operations: initialization, selection, crossover, and mutation [34]. In this paper, we define a chromosome as a feasible mapping of a VNR. Each chromosome includes a node chromosome and an associated link chromosome. A population is an elite set of up to $\mathcal{M}$ chromosomes. A node chromosome is a feasible mapping of all virtual nodes of a VNR where a gene is a feasible mapping of a single virtual node. A link chromosome is a feasible mapping of all virtual links of a VNR where a gene is a feasible mapping of a single virtual link. Towards the idea of joint node and link mappings based on GA algorithm in this paper, when an online VNR arrives, GA will search for an optimal embedding solution including both node and link mappings. Population initialization randomly generates node chromosomes and then associated link chromosomes through VNoM and VLiM. During this process, heuristic conciliation strategy is deployed to remap virtual nodes that cause the failures of VLiM. Then, selection selects two random node chromosomes to become parents which are utilized in evolution processes. These parents will swap their genes starting from a random crossover point to generate new children. Eventually, one of the children is selected for mutation process and one random gene of this child is then replaced by a new random one to produce a new child. The population will update those children while maintaining elite chromosomes by substituting worst existing chromosomes based on fitness values. Evolution is continued with new selection and this procedure will run for several iterations in aimed at obtaining the optimal mapping solution.

Lately, parallel computing is a promising paradigm to efficiently deal with the complicated issues with considerable time saving and low cost guarantees thanks to concurrent-enabled deployment. In practise, GA can be fundamentally recognized as a parallel searching mechanism [35] with non-mutual dependency amongst several exclusively feasible solutions. To resolve the aforementioned problem of GA algorithm, we propose a distributed parallel implementation based on GA algorithm in next section.

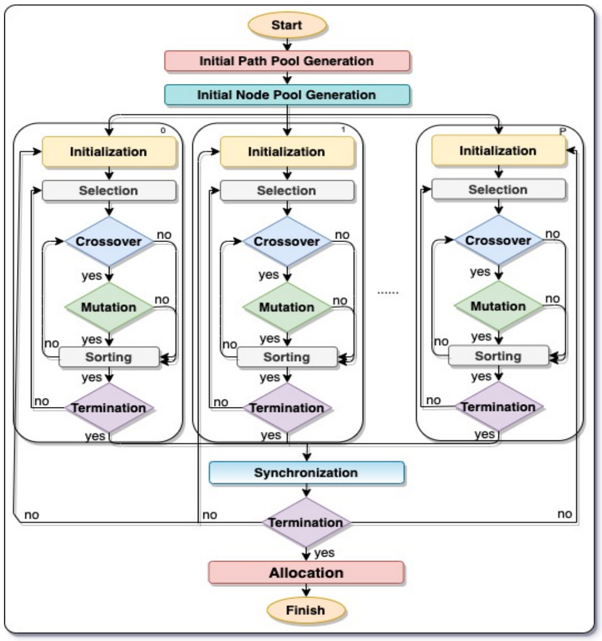


Fig. 3: *Parallel operation scheme*

B. Distributed parallel GA-based algorithms

Distributed and parallel computing has been practically proven as an efficient paradigm to deal with enormous and complicated problems with guaranteed costs and less operational time by concurrency support. Our proposed parallel GA-based scheme in this paper is presented in Fig 3. Every working machine in this scheme independently carries out a GA algorithm to approach as many feasible solutions of a VNR as it can with a specific number of iterations. The best-matching result is determined amongst those distributed and parallel machines. It is noted that when node mappings are changed, the link mappings altered accordingly. Operational details of our proposed scheme are described in Algorithm 1. Lines [5-15] express initial path and node pool generations respectively, while lines [16-20] are the VNE procedures. More details about GA algorithm can be found in Algorithm 2 where lines [7-22] are associated with the initial population generation and lines [23-46] include the evolution processes.

Algorithm 1 Distributed parallel GA-based algorithm

```
 1: Input:
 2:     + Online $G_i^v$
 3: Output:
 4:     + Near-optimal mappings for $G_i^v$
 5: function INITIAL PATH POOL GENERATION
 6:     Construct $\mathcal{O}_p^s$ using k-shortest path algorithm based on
        hop-count
 7: end function
    ▷ Create an initial node pool $\mathcal{O}_{n_i}^s$ for $G_i^v$
 8: function INITIAL NODE POOL GENERATION
 9:     $n_i^{v_m} = argmin\ c(n_i^v)$
10:     for each $n^s \in N^s$ do
11:         if $R_N(n^s) \geq c(n_i^{v_m})$ add $n^s$ to $\mathcal{O}_{n_i}^s$.
12:     end for
13:     if $|\mathcal{O}_{n_i}^s| \geq |N_i^v|$ goto VIRTUAL NETWORK EMBED-
        DING
14:     else reject $G_i^v$
15: end function
16: procedure VIRTUAL NETWORK EMBEDDING
    ▷   Implement genetic algorithm in distributed parallel
        machines
17:     call Algorithm 2
    ▷   Synchronize the achieved incumbents among parallel
        machines
18:     Select the best feasible solution across all machines
        based on fitness value
19:     if none exist reject $G_i^v$
20:     else accept $G_i^v$ and update substrate network resources
21: end procedure
```

C. Node Mapping Algorithm

In our proposed joint node-link mapping algorithms where the GA algorithm is the core, the node mapping of a given VNR simply works in a heuristic way. Population in GA's operations consisting of several chromosomes is generated in random manner. A VNR includes a set of virtual nodes connected with several virtual links. Accordingly, there are separate node mappings and link mappings. The later is dependent on the former, so if the node mappings are changed, the link mappings are altered correspondingly. We define a chromosome $\mathcal{C}_f$ as a mapping solution for all virtual nodes in a given VNR.

Algorithm 2 Genetic Algorithm with Conciliation Strategy at each parallel machine

```
 1: Input:
 2:     + Online $G_i^v$
 3:     + $\mathcal{O}_p^s$ and $\mathcal{O}_{n_i}^s$
 4: Output:
 5:     + The mapping solution for $G_i^v$
 6: procedure GENETIC ALGORITHM OPERATIONS
    ▷ Generate initial population
 7:     p =0
 8:     for $m = 1$ to $\mathcal{M}$ do
    ▷       Generate a node chromosome with $|N_i^v|$ genes
 9:         for $n = 1$ to $|N_i^v|$ do
    ▷           Try to map virtual node n to a randomly selected
                node in $\mathcal{O}_{n_i}^s$ with up to Q trials
10:             for $q = 1$ to Q do
11:                 Map virtual node n to a randomly selected node
                    in $\mathcal{O}_{n_i}^s$
12:                 if feasible goto 15
13:             end for
14:             goto 22
15:         end for
    ▷       Generate a link chromosome with $|L_i^v|$ genes
16:         for $l = 1$ to $|L_i^v|$ do
17:             Try to map virtual link $l$ to a selected path in $\mathcal{O}_p^s$
    ▷           GA-RAN, GA-SEQ, GA-STP are different in path
                selection
18:         end for
19:         if some link mappings are infeasible call Algorithm 3
20:             if Algorithm 3 returns false goto 22
21:         $p = p + 1$ and add chromosome to population
22:     end for
    ▷ Evolution process
23:     if $p > 1$ then
24:         for $e = 1$ to maxIterations do
25:             if $ranNum \in (0, 1) < p_c$ then
26:                 Randomly select two parents
27:                 Conduct crossover operation
28:                 if both children are feasible then
29:                     One of children is randomly selected for
                        mutation
30:                     $p = p + 1$ and add the one produced better
                        fitness values to population
31:                 else
32:                     if only one child is feasible
33:                         $p = p + 1$ and add the one to population
34:             if $ranNum \in (0, 1) < p_m$ then
35:                 if none of children in crossover are feasible then
36:                     Randomly select one of parents for mutation
37:                 end if
38:                 Conduct mutation operation
39:                 if the mutated child is feasible then
40:                         $p = p + 1$ and add the one to population
41:         end for
42:         if $p > \mathcal{M}$ eliminate the ones with lower fitness values
43:     else if $p = 1$ then the current mapping will be the final
44:     else
45:         reject $G_i^v$ for this machine
46:     end if
47: end procedure
```

Algorithm 3 Heuristic Conciliation Algorithm

```
 1: Input:
 2:     A set of failed virtual links after VLiM $\mathcal{F}_i^v$
 3: Output:
 4:     A feasible chromosome or false
 5: procedure HEURISTIC CONCILIATION STRATEGY
 6:     Construct a map $M_p$ with each element representing the
        number of failed virtual links associated with one virtual
        node.
 7:     Create a map $M_n$ with each element representing the degree
        of a virtual node.
    ▷ Select virtual nodes for remapping
 8:     Initialize a remapping list $\mathcal{E}_i^r$
 9:     for each failed virtual link $(u^v, k^v) \in \mathcal{F}_i^v$ do
10:         if virtual node $u^v$ or $k^v$ not in $\mathcal{E}_i^r$ then
11:             if $M_p[u^v] > M_p[k^v]$
12:                 add $u^v$ into $\mathcal{E}_i^r$
13:             else if $M_p[u^v] < M_p[k^v]$ then
14:                 add $k^v$ into $\mathcal{E}_i^r$
15:             else                                   ▷ $M_p[u^v] = M_p[k^v]$
16:                 if $M_n[u^v] > M_n[k^v]$ then
17:                     add $k^v$ into $\mathcal{E}_i^r$
18:                 else
19:                     add $u^v$ into $\mathcal{E}_i^r$
20:     end for
21:     return $\mathcal{E}_i^r$
    ▷ Create a new chromosome with at almost $\mathcal{R}$ attempts
22:     for $r = 0$ to $\mathcal{R}$ do
    ▷       Generate a new chromosome with $\mathcal{E}_i^r$
23:         for each virtual node $n_i^v \in \mathcal{E}_i^r$ do
    ▷           Try to map virtual node $n_i^v$ to a randomly selected
                node in for up to $\mathcal{Q}$ trials
24:             for $q = 1$ to $\mathcal{Q}$ do
25:                 Map virtual node $n_i^v$ to a randomly selected node
                    in $O_{n_i}^s$
26:                 if feasible break
27:             end for
28:             if none of the trials are feasible goto 36
29:         end for
    ▷       Generate a link chromosome with associated virtual links
30:         for $l_r$ in the set of affected virtual links do
31:             Try to map virtual link $l_r$ to a selected path in $\mathcal{O}_p^s$
    ▷           GA-RAN, GA-SEQ, GA-STP are different in path
                selection
32:             if any link mapping is infeasible goto 35
33:         end for
34:         return new chromosome goto 37
35:     end for
36:     return false
37: end procedure
```

Each gene Φ_f^j is associated with a mapping of a virtual node, where f and j indicate the f^{th} chromosome and j^{th} virtual node in the VNR respectively. We equally consider all virtual node mappings. Therefore, their order in a chromosome can be assigned in an arbitrary manner. Once assigned, all the chromosomes will follow the same order. To enhance the capability and reduce execution time, we initially generate a feasible node pool where all eligible substrate nodes meeting the least resource demand of the VNR are collected. Each virtual node in a VNR is consecutively mapped by a substrate node which is randomly selected in the initial node pool, that must meet resource requirements (Eq. (1)-(4)) to become a feasible node mapping. We assume that mapping virtual nodes randomly can prevent the premature convergence problem of GA algorithm.

D. Link Mapping Algorithms

When node chromosome is successfully formed, we then generate the according link chromosome using different link mapping algorithms, in which each gene of the link chromosome is a link mapping of a virtual link in the VNR that is embedded onto a substrate path through VLiM. VLiM can be reduced to unsplittable flow problem which has been widely recognized as $\mathcal{NP}$-hard [3]. To solve VLiM, we build a path database using the shortest path method (e.g., Dijkstra's algorithm) with respect to the hop-count factor, which can be completely constructed prior to the arrival of online VNRs. This database is called initial path pool generation in Fig. 3. This process is possible since we argue that substrate paths for each pair of physical nodes in a SN can be determined in advance due to the fact that topology of a SN is most likely static. In this paper, we propose three virtual link mappings with different searching mechanisms for VLiM stage.

1) Random Path Searching Algorithm (GA-RAN): For a virtual link request in a VNR, the path searching algorithm randomly selects a substrate path from the path database based on the information of node mappings. This path must pass a validity check Eq. (5)-(7) to become a feasible path. If the path fails to pass the resource checking process, the algorithm attempts to find another one in the database until at least one feasible path is achieved. If there is no feasible path found in the path database, the algorithm returns failed to the virtual link request. In that case, the previous node mappings should be reconsidered. Each virtual link in the given VNR is sequentially searching for a feasible substrate path in that way until all virtual links are processed.

2) Sequential Single Path Searching Algorithm (GA-SEQ): Instead of searching the path in random, the sequential path searching mechanism searches a potential path for a virtual link request from the first path in the database until the last one sequentially. Whenever there is a substrate path passing the validity check, it becomes feasible, and then the searching algorithm will immediately stop there. In fact, the erected path database is relied on the hop-count, so the first path in database has the lowest hop-count feature. In contrast with GA-RAN, GA-SEQ goes through the path database by a sequential path searching. After the whole path database is inspected, if there is no feasible path achieved, the algorithm returns failed similar to GA-RAN. All virtual links in the VNR are handled in the same way. All link mappings that are specified as "failed" will be then reprocessed in the conciliation algorithm. Due to the simple and straightforward features of the sequential single path searching mechanism, we expect that its execution time of this algorithm is better than those of other proposed algorithms.

3) Shortest Path Algorithm (GA-STP): Different from the aforementioned path searching mechanisms, we deploy k-shortest path algorithm as an approximation approach so as to optimize link delay and bandwidth consumption of a successfully mapped virtual network. Based on the information of node mappings, GA-STP maps each virtual link of the VNR to a single physical path. The shortest path algorithm searches

k-shortest paths by increasing the value of k until a feasible path that has sufficient bandwidth to embed the corresponding virtual link is found. The time complexity of the algorithm increases with the value of k. For an efficiency of computation and network resource usage as well as reducing link delay which is associated with the distance, the value of k should be kept minimal, so we choose $k = 1$ in this paper. In fact, we prune all links that do not have sufficient bandwidth to support the virtual link request, and then choose the shortest-distance path in the remaining graph. In case there is no feasible path found, the according virtual link is marked as failed. This link method is able to efficiently balance the network load among the links and effectively utilize the network resources. By applying the conciliation strategy, infeasible link mappings can be revised by revisiting the current node mappings smartly, which requires to automatically explore new link mappings due to the dynamic changes of node mappings. It is clearly not subject to the value of k because the explored new link mappings are associated with different node mappings rather than the k-shortest paths associated with the same node mappings. In GA-STP, we set k = 1 to highlight this effect.

We expect that our proposed joint node-link mapping assisted by the shortest path mechanism is able to achieve the best performance compared to its counterparts, especially the average link delay metric. A desirable outcome towards the delay metric is beneficial for the virtual network requests that strictly require sensitive delay requirements (e.g., IoT applications).

E. Conciliation Construction

When a node chromosome is determined, various path searching methods as described in V-D are applied to form the according link chromosome. If they successfully find out link mappings for the link chromosome, a feasible solution for the whole VNR has been achieved. However, this is an ideal scenario. In reality, there might have a situation that link mapping algorithms cannot find any feasible substrate path for a virtual link request due to network congestion. Generally, the pair of virtual nodes that form such virtual link needs to be remapped. In worse cases, several virtual link requests fail to be embedded, so it is desired to have an efficient mechanism to intelligently remap these requests concerning the minimal number of virtual nodes associated with the failed virtual links that should be revisited.

Let's take a real VNR with 7 virtual nodes and 6 virtual links as demonstrated in Fig. 4 with several practical scenarios for an example. Fig 4a is an accepted solution since all virtual nodes and links have been successfully found their feasible mappings. Next figure illustrates a simplest failure with a single failed virtual link mapping from $A - G$. In this case, the substrate nodes h and n, node mappings of virtual nodes A and G, respectively, need to be remapped. If n is selected, we would remap only one virtual link $A - G$; otherwise, we have to remap three virtual links $\{A - G, A - F, A - D\}$. Fig. 4c complicates the same problem a bit with two failed virtual links $\{A - G$ and $A - D$ $\}$. We possibly have three options for remapping including substrate nodes {h, k and n}. Each selection would give the same number of remapped virtual links, but h should be chosen instead of k and n because least virtual nodes need to be reconsidered. Similarly, h should be selected for remapping in Fig. 4d, even there are three failed link mappings including {A - G, A - F and A - D}. If these all already-mapped virtual nodes or virtual nodes {D, F and G} are revisited, we need to remap at least five virtual links. Otherwise, when the substrate node h, a node mapping solution of virtual node A, is reconsidered, the remapped virtual links could be only three. In Fig. 4e, substrate nodes k/j, l and n should be remapped, whereas physical nodes j and i can be re-embedded in Fig. 4f due to least remapping.

Inspired by this idea, the heuristic conciliation algorithm is applied to handle a set of infeasible link mappings for generating each chromosome. When we remap virtual nodes associated with the failed virtual links, we will remap them following an order based on the importance of each virtual node. M_p and M_n are designed to evaluate the importance of each virtual node based on the impacts of the virtual node. M_p is a map with each element in the map representing the number of failed virtual links associated with one virtual node. M_n is the map with each element representing the degree of a virtual node. Details of the heuristic conciliation algorithm is described in Algorithm 3 where lines [08-21] attempt to quantify potential nodes for re-mapping.

For better understanding, let us take an example in Fig. 4f for our remapping strategy applied on the SN with the topology that is demonstrated in Fig. 2. The VN is shown at right corner of Fig. 5 while the SN can be abstracted on the left side. We have three failed virtual links ($F - E, D - C$ and $C - B$). After applying the conciliation strategy, it is merely required to remap virtual nodes C and E instead of all virtual nodes in which old node mappings ($C \to j, E \to l$) are now remapped to new mappings ($C \to s, E \to p$). As a result, all virtual links are mapped successfully.

F. Working machine

As illustrated in Fig 3, each paralleled machine is independently running a GA algorithm which is consisted of four major operations: population initialization, selection, crossover and mutation.

Population Initialization: A single machine starts with a population initialization. Denote $\mathcal{M}$ as a set of chromosomes forming a population. We have separate node population (27) and link population (28). Each node chromosome includes $\mathcal{N} = |N_i^v|$ genes that represent a potential mapping solution for all virtual node requests in a VNR, where each gene is the node mapping of a virtual node. An initial node population $\mathcal{P}_n$ ($\mathcal{M}$x$\mathcal{N}$ size) at each machine is generated as described in Section V-C.

$$\mathcal{P}_n = \begin{bmatrix} \mathcal{C}_1^n \\ \mathcal{C}_2^n \\ \vdots \\ \mathcal{C}_f^n \\ \vdots \\ \mathcal{C}_{\mathcal{M}}^n \end{bmatrix} = \begin{bmatrix} \Phi_1^1 & \cdots & \Phi_1^j & \cdots & \Phi_1^{\mathcal{N}} \\ \Phi_2^1 & \cdots & \Phi_2^j & \cdots & \Phi_2^{\mathcal{N}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \Phi_f^1 & \cdots & \Phi_f^j & \cdots & \Phi_f^{\mathcal{N}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \Phi_{\mathcal{M}}^1 & \cdots & \Phi_{\mathcal{M}}^j & \cdots & \Phi_{\mathcal{M}}^{\mathcal{N}} \end{bmatrix} \quad (27)$$

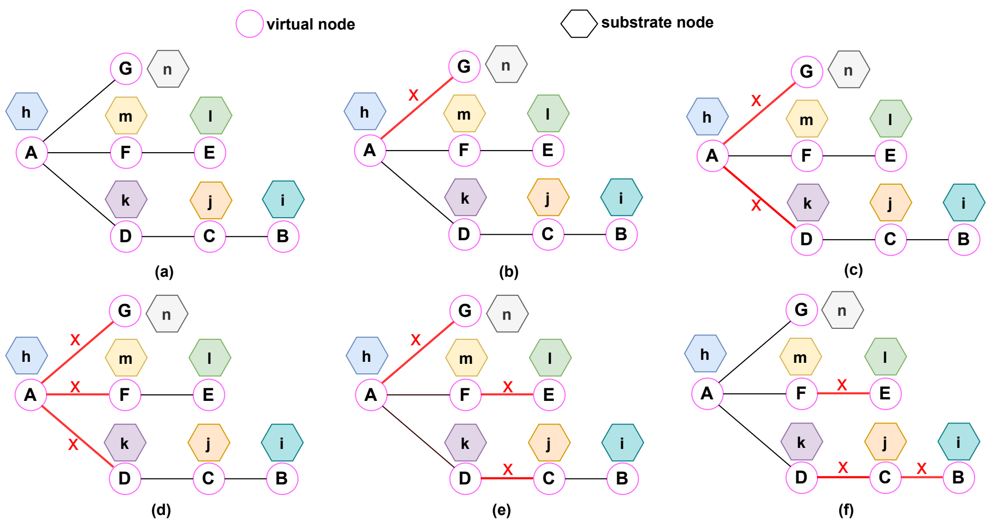


Fig. 4: *Examples of infeasible virtual link mappings*

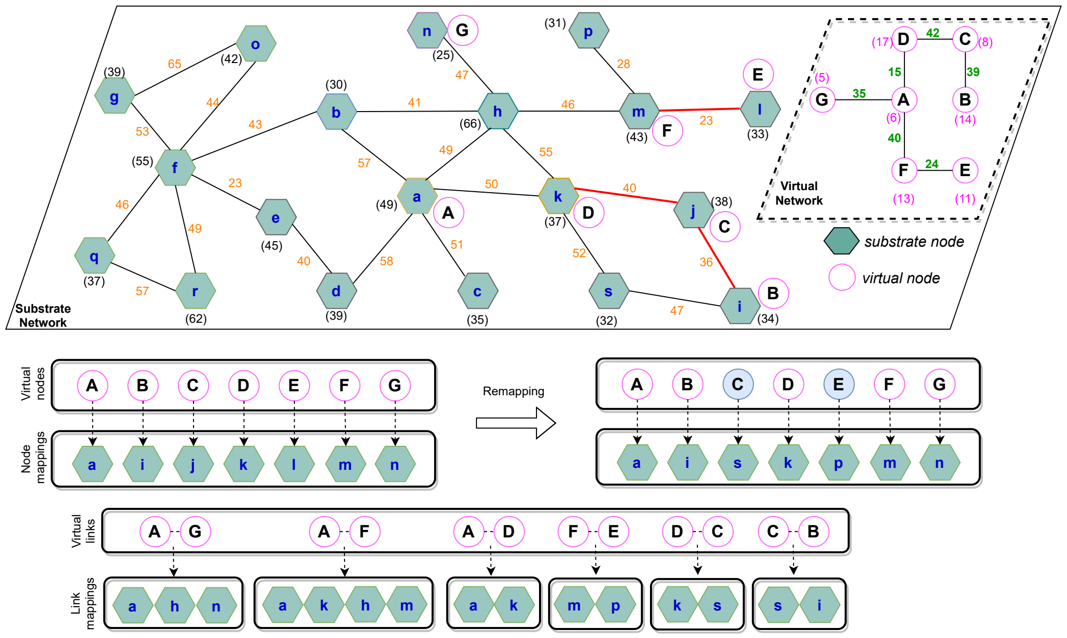


Fig. 5: *Examples of Conciliation and remapping*

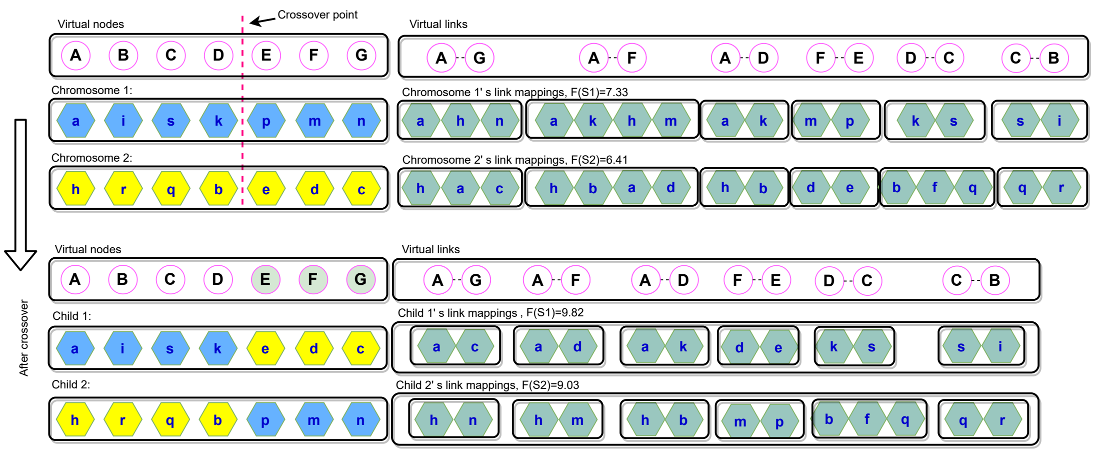

Crossover point
Virtual nodes
Virtual links
Chromosome 1:
Chromosome 1' s link mappings, F(S1)=7.33
Chromosome 2:
Chromosome 2' s link mappings, F(S2)=6.41
After crossover
Child 1:
Child 1' s link mappings , F(S1)=9.82
Child 2:
Child 2' s link mappings, F(S2)=9.03

Fig. 6: *Crossover operation*

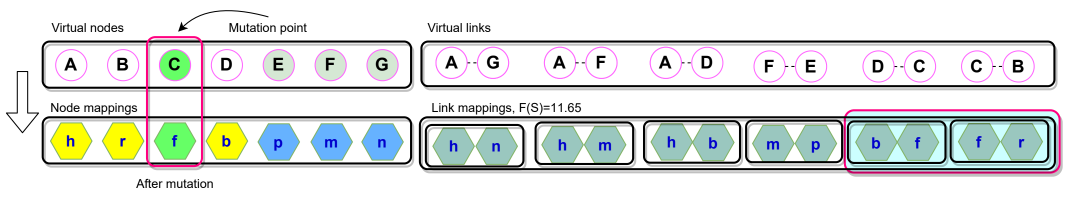


Fig. 7: *Mutation operation*

$$\mathcal{P}_e = \begin{bmatrix} \mathcal{C}^e_1 \\ \mathcal{C}^e_2 \\ \vdots \\ \mathcal{C}^e_f \\ \vdots \\ \mathcal{C}^e_{\mathcal{M}} \end{bmatrix} = \begin{bmatrix} \Psi^1_1 & \cdots & \Psi^j_1 & \cdots & \Psi^{|L^n_i|}_1 \\ \Psi^1_2 & \cdots & \Psi^j_2 & \cdots & \Psi^{|L^n_i|}_2 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \Psi^1_f & \cdots & \Psi^j_f & \cdots & \Psi^{|L^n_i|}_f \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \Psi^1_{\mathcal{M}} & \cdots & \Psi^j_{\mathcal{M}} & \cdots & \Psi^{|L^n_i|}_{\mathcal{M}} \end{bmatrix} \tag{28}$$

For each node chromosome, we generate a corresponding link chromosome representing a potential mapping for all virtual links in a VNR, where each gene is the link mapping of a virtual link request based on the information of node mappings. The generation of link chromosomes can be found in Section V-D, while infeasible link mappings produced by this process are dealt with by the conciliation strategy as described in Section V-E. When reached $\mathcal{M}$ chromosomes for both node and link mappings, our population is "officially" established.

In fact, we cannot always generate $\mathcal{M}$ chromosomes when the network becomes more and more congested. Therefore, the search for $\mathcal{M}$ chromosomes will be stopped after a preset number of iterations and the remaining GA operations will be processed with the available chromosomes. We need at least two parents to produce children. If there is only one chromosome available after the iteration process, the GA process will be finished with the chromosome returned as the only feasible solution. As mentioned, we are focusing on node mappings, whereas link mappings will be changed according to any changes of node mappings. For simplification, we therefore concentrate on generations of node mappings in next GA operations.

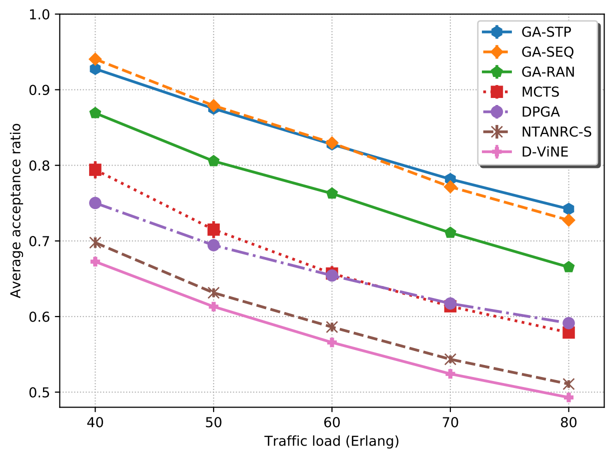
New generations In this paper, we randomly select chromosomes to be parents for generating their children. Selected parents produce new generations as a result of crossover and mutation operations, which consequently makes the mapping solutions evolved after the number of iterations. After those operations, the major problem is seeking feasible physical paths regarding to changes of new nodes. Similar to Section V-E, the path searching methods are implemented to explore link mapping solutions for new generations towards new pairs of substrate source-destination nodes. If no feasible paths found on new generations, they will be discarded.

The chromosome population we maintain has up to $\mathcal{M}$ chromosomes. Whenever new children are added to the population, we eliminate the ones with lowest fitness values to make sure that the population has at most $\mathcal{M}$ chromosomes. Therefore, the population itself is an elite set. By randomly selecting from the population, we attempt to maintain a balance between exploration and exploitation. When the iteration of generations is terminated, we select the best ones among all generations as our final solution. This also guarantees the best ones are selected. We have added Fig. 12 to show the convergence process.

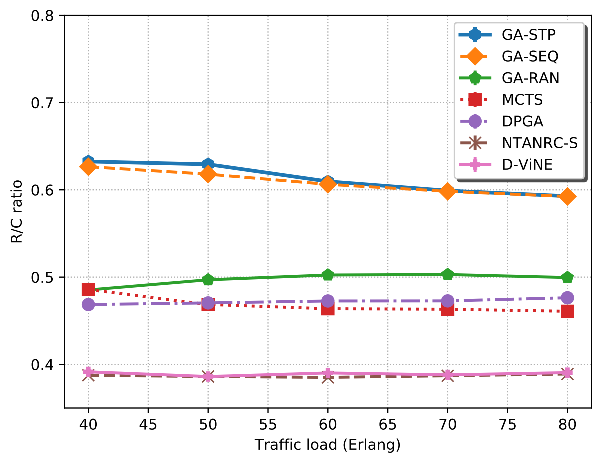
$$\mathcal{P}_n = \begin{bmatrix} \mathcal{C}^n_1 \\ \vdots \\ \mathcal{C}^n_s \\ \vdots \\ \mathcal{C}^n_r \\ \vdots \\ \mathcal{C}^n_{\mathcal{M}} \\ \mathcal{C}^n_{\mathcal{M}+1} \\ \mathcal{C}^n_{\mathcal{M}+2} \end{bmatrix} = \begin{bmatrix} \Phi^1_1 & \cdots & \Phi^{j^c}_1 & \Phi^{j^c+1}_1 & \cdots & \Phi^{\mathcal{N}}_1 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ \Phi^1_s & \cdots & \Phi^{j^c}_s & \Phi^{j^c+1}_s & \cdots & \Phi^{\mathcal{N}}_s \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ \Phi^1_r & \cdots & \Phi^{j^c}_r & \Phi^{j^c+1}_r & \cdots & \Phi^{\mathcal{N}}_r \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ \Phi^1_{\mathcal{M}} & \cdots & \Phi^{j^c}_{\mathcal{M}} & \Phi^{j^c+1}_{\mathcal{M}} & \cdots & \Phi^{\mathcal{N}}_{\mathcal{M}} \\ \Phi^1_s & \cdots & \Phi^{j^c}_s & \Phi^{j^c+1}_r & \cdots & \Phi^{\mathcal{N}}_r \\ \Phi^1_r & \cdots & \Phi^{j^c}_r & \Phi^{j^c+1}_s & \cdots & \Phi^{\mathcal{N}}_s \end{bmatrix} \tag{29}$$

Crossover: In this operation, parental chromosomes are combined in order to create new offspring for next generations. $\mathcal{C}_s$ and $\mathcal{C}_r$ represent the selected chromosomes where their indexes within the initial population are s and r respectively. j^c is the crossover point which is randomly chosen between any positions within $\mathcal{N}$ length. We denote $\mathcal{C}_{(\mathcal{M}+1)}$ and $\mathcal{C}_{(\mathcal{M}+2)}$ as new generated chromosomes. In crossover, offspring is typically produced by exchanging genes between the selected parents beginning from the random crossover point j^{c+1} to the end of chromosomes as demonstrated in (29). At this stage, node mappings have been changed, link mappings are revisited by the path searching methods accordingly as described in Section V-D. In this paper, a physical node cannot host two virtual nodes with the same VNR. If the crossover or mutation leads to nodes mapped to the same physical node, we will treat the mapping as infeasible and discard it. Suppose we select two chromosomes from the population as parents. As depicted in Fig. 6, the random crossover point equals to 4, the node mappings of virtual nodes E, F and G will be swapped between two parents in order to generate new offspring. After new node mappings are determined, we need to examine their corresponding link mappings. We can recognize that new link mappings are much better than the previous ones, which means that our children get evolved.

Mutation: This operation typically adopts a modification on a parent to generate a new offspring. Mutation samples the broad solution space and improves the searching efficiency, preventing solutions from falling into the local optima. In this paper, one of the feasible children created in Crossover is randomly selected for mutation, and a random gene of this child is then replaced by a new gene to produce a new offspring. The

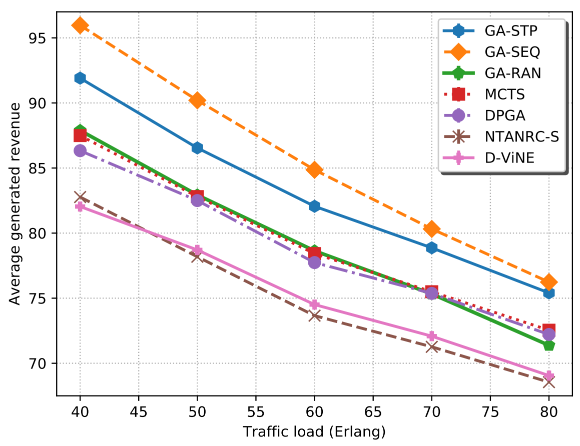


(a)

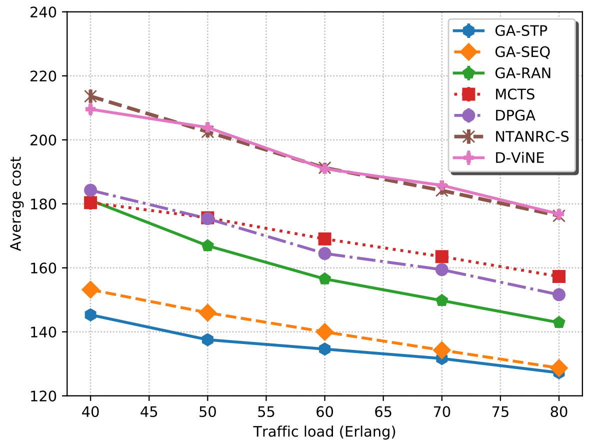


(b)

Fig. 8: **(a)** Average acceptance Ratio **(b)** Average revenue to cost ratio



(a)

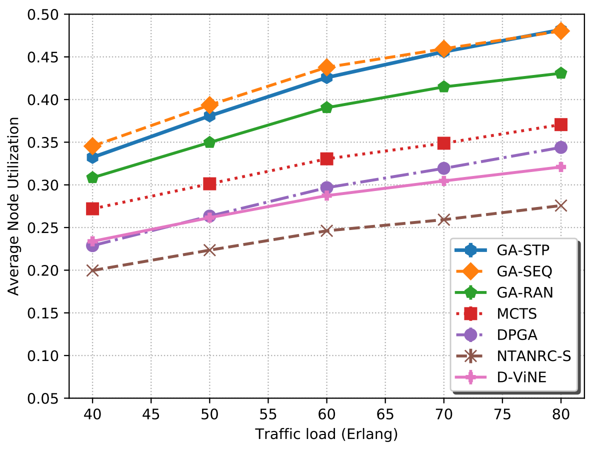


(b)

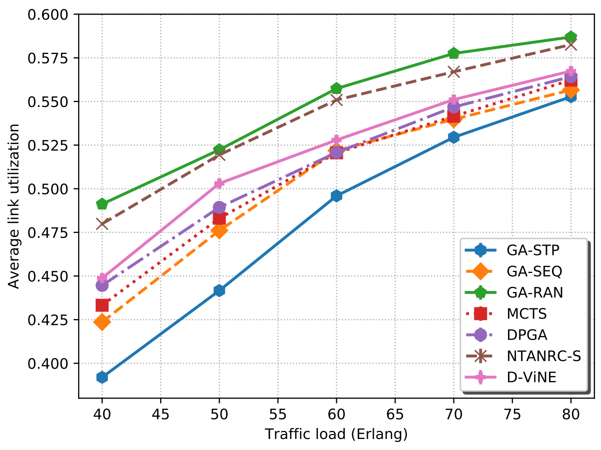
Fig. 9: **(a)** Average generated revenue **(b)** Average embedding cost

gene used for replacement must explicitly meet the resource constraints. However, if none of children in Crossover are feasible, one of parents is selected for mutation in random as illustrated in Algorithm 2. Similar to crossover, path searching mechanisms are implemented to find new link mappings due to node mappings changed. If feasible, new generation is updated into population. Let denote j^m as a random mutation point and $\Phi_{r'}^{j^m}$ is new gene that replaces the existing one in $\mathcal{C}_{(\mathcal{M}+1)}$. In this operation, new embedding solution $\mathcal{C'}_{(\mathcal{M}+1)}$ after replacement is presented as $\mathcal{C'}_{(\mathcal{M}+1)} = [\Phi_s^1 \cdots \Phi_{r'}^{j^m} \cdots \Phi_s^{\mathcal{N}}]$. As illustrated in Crossover, we achieved new children which are better than their parents. Suppose the second child is randomly chosen for the mutation, where the mutation point is selected in random, equal to 3 as depicted in Fig. 7. Old node mapping $(C \to q)$ is randomly replaced by $(C \to f)$, which results in new link mappings for virtual links ({D-C} and {C-B}). Specifically, substrate paths $(b \to f \to q)$ and $(q \to r)$ are now substituted by $(b \to f)$ and $(f \to r)$ respectively. Both new children are feasible and better than their parents based upon improved fitness values shown on the top right of each subfigures of Fig. 6 and Fig. 7, so they will be updated into the population.

In fact, maintaining a proper balancing between exploitation and exploration is essential in GA. Crossover rate p_c is in a range of [0-1] that is normally kept in high values to provide a better exploitation on the current population. Mutation rate p_m is usually smaller than the crossover rate. Selecting an appropriate value for the mutation rate is challenging since a high mutation rate would increase the possibility of exploring more areas in the searching space, but could prevent GA from converging to any optimal solution. In contrast, small mutation rate might cause a problem of premature convergence that traps the population in local optima. By preferring high efficiency of GA while keeping a trade-off between exploitation and

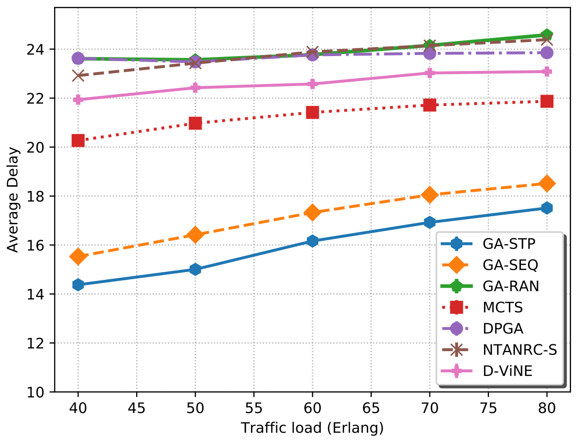


(a)

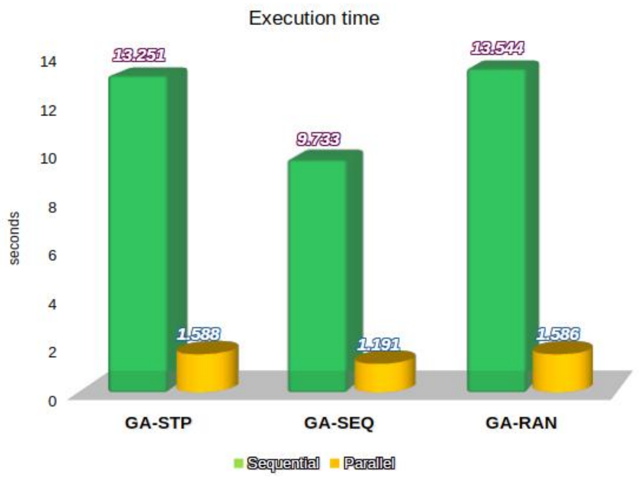


(b)

Fig. 10: **(a)** Average node utilization **(b)** Average link utilization



(a)



(b)

Fig. 11: **(a)** Average delay **(b)** Execution time between our algorithms

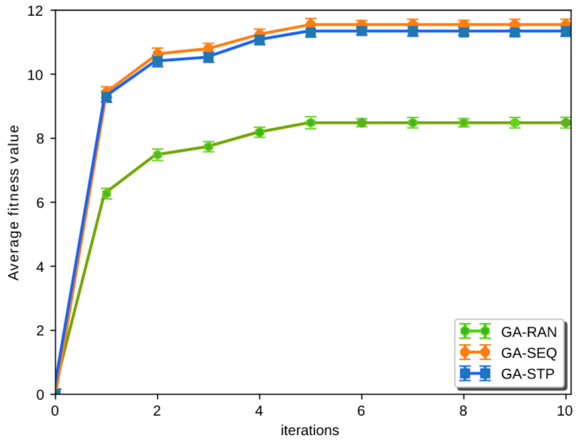


Fig. 12: *Fitness Function Convergence*

exploration, we set $p_c = 0.9$ and $p_m = 0.2$ in this paper. As depicted in Fig. 12, our GA algorithms converged smoothly and stably. Our numerical results have also shown better performances than other algorithms.

G. Sorting and Terminations

Sorting process selects the best mapping amongst the achieved feasible solutions determined by their fitness values. Its outcome is then sent to the synchronization process for a global ranking. In fact, parallelism is associated with several concurrent processes where each may finish its assigned task at different time. Waiting until the last process accomplishing its job without guaranteeing an expected outcome is indeed troublesome. A total of 16 working machines were used. For each working machine, we set M = 10. For each generation of each working machine, two chromosomes are selected as parents. After the crossover process, two children will be generated. One of the two will be randomly selected to do the mutation process. Then the one which is not selected and the one which is generated after mutation will be added to the pool with up to $\mathcal{M}$ chromosomes. If more than $\mathcal{M}$ chromosomes are available, the ones with least fitness values will be deleted to maintain the pool as an elite set. The maximum number of iterations (generations) was set to 10. To keep the trade-off between operation time and performance, the iteration process will be terminated earlier if the best fitness value does not change for t = 4 consecutive iterations.

H. Synchronization and VN allocation

In this step, the best VNE solution of the corresponding VNR is determined by globally ranking the VNE solutions received from worker nodes, based on highest achieved FF values. As a result, if accepted the given VNR is allocated to the corresponding SN following the information of node and link mappings. The last step is updating residual network resources.

I. Execution time analysis

Due to the lower cost of computing hardware recently, parallel algorithms can be beneficially exploited to tackle intricate computational tasks. As a result, we propose a distributed parallel GA framework to deal with the online VNE problem in this paper. We measure the execution time of the proposed framework on two different modes: sequential and parallel. In the former, the time complexity increases linearly since the total time equals the sum of the execution time of each worker machine. In contrast, the latter's total execution time is counted by the worker machine that accomplishes its assigned task at the latest. Cramer-Chernoff method and Jensen's inequality can approximate the total execution time in parallel mode as detailed in [27]. Accordingly, the proposed distributed parallel framework can rapidly improve the time complexity from linear to logarithmic scale. Interested readers may refer to [27] further theoretical analysis.

VI. Performance Evaluation

We compare our proposed GA-RAN, GA-SEQ and GA-STP algorithms with several state-of-the-art VNE competitors: DPGA [9], NTANRC-S [19], D-ViNE [7] and MCTS [28] on several evaluation metrics comprising average acceptance ratio, average revenue and cost, average R/C, average node and link utilization and finally average delay. These are the most popular and important evaluation criteria for evaluating the efficiency of any VNE algorithm. We select D-ViNE since it is the best performance algorithm in [7] which is widely known as one of the most highly-cited papers in VNE. Moreover, NTANRC-S, performed best in [19], is the latest representative of the node ranking methods that exploits multiple topology attributes and the global network resources for VNoM in VNE. Recently, our work in [9] has confirmed that VLiM contributes a critical role in approaching an efficient solution for VNE problem. Finally, MCTS is the reinforcement learning-based algorithm that deploys Monte Carlo Tree Search to explore the action space [28]. The proposed GA-based algorithms in this paper are expected to compete those VNE algorithms in all performance metrics. Our simulation was performed on a Ubuntu 20.04.2 LTS (Focal Fossa) 64-bit platform with 16 GB memory and Intel®Core™ i5-6200 CPU @2.30GHzx4.

TABLE III: *Compared Algorithms*

Notation	Description
GA-STP	Joint node-link GA-based algorithm assisted by shortest path method
GA-SEQ	Joint node-link GA-based algorithm assisted by sequential single path searching
GA-RAN	Joint node-link GA-based algorithm assisted by random path searching
MCTS	Monte Carlo Tree Search with BFS algorithm for VLiM [28]
DPGA	Greedy node mapping with distributed parallel GA-based algorithm for link mapping [9]
NTANRC-S	Network topology attributes and network resource-considered algorithm for node mapping with the shortest path-based link mapping [19]
D-ViNE	Deterministic rounding-based approach obtaining a LP relaxation of MIP for node mapping with shortest path method for link mapping [7]

A. Simulation setup

A discrete-event simulator is developed to evaluate the proposed joint node-link mapping solutions with the evaluation parameters following [7]. We extensively generate SNs and VNs using the most common GT-ITM topology generator [36]. SNs, adopting Waxman model with $\alpha = 0.5$ and $\beta = 0.2$ where α and β indicate the maximal edge probability and edge length respectively, are consisted of 50 nodes randomly placed on a 25×25 Cartesian plane and 140 edges in average. Their capacity including CPU and bandwidth is uniformly distributed between 50 and 100 units. Similar to the previous research [7], VNRs are configured to follow a Poisson process dynamically arriving in the network with an average λ rate ranging from 4 to 8 VNs per 100 time units. Each VNR has an exponentially distributed lifetime with an average value of $\mu = 1000$ time units. The load of VNRs can be indeed measured by $\frac{\lambda}{\mu}$ Erlangs [28]. In addition, each VNR's node size is uniformly distributed between 2 and 10. The resource requirements of VNRs are uniformly distributed between 0 to 20 for virtual nodes and 0 to 50 for virtual link requests. Following [12], we set $w_b = w_n = 1$ in this paper. For simulation purposes, we deploy three different SNs and two sets of heterogeneous VNRs, where each includes five subsets of VNRs ranging from $2,000$ up to $4,000$ VNRs. In this paper, we set out $Q = 2 \times M, \mathcal{Q} = \mathcal{R} = 2 \times (|\mathcal{F}_i^v| + |\mathcal{E}_i^r|)$.

The evaluation results were generated by averaging over three different SNs and two replications of VNRs with each replication running $50,000$ time units under each traffic load and each SN. This setting is the same as the parameters in [12]. We have run all comparisons in the same setting. All figures are plotted with mean values and 95% confidence intervals (CIs). In fact, error bars are too tiny, confirming that our evaluation outputs are extremely reliable. For better visualization, we plot figures with various colours and different markers. Because the $50,000$ time units are 50 times longer than the average lifetime of a VN, it provided roughly 50 independent samples per replication per substrate network. Considering the 2 replications and 3 different SNs, we have about 300 independent samples under each load condition, that is why we can get very good CIs.

B. Discussions on Performance Results

In this section, we discuss on the simulation results between our proposed VNE algorithms and the compared algorithms. Then, a comprehensive analysis in performance between the joint node-link algorithms themselves is also presented. Fig. 8 presents the average VNR acceptance ratio and the average R/C, while Fig. 9 shows the average generated revenue as well as the average embedding cost. They are by some means sufficient to confirm the efficiency and effectiveness of our joint node-link VNE solutions. The solid performance of the proposed algorithms is intensified in Fig. 10 where the average network utilization including nodes and links is illustrated. Fig. 11 additionally highlights the advantage of our solutions on average delay, and a comparison of execution time between the proposed algorithms running on sequential and parallel modes is also demonstrated.

1) Joint node-link algorithms vs State-of-the-art VNE algorithms: Fig. 8a illustrates the average VNR acceptance ratio which is one of the most important metrics to evaluate the embedding capability of different VNE algorithms. It is observed that the average VNR acceptance ratio of all algorithm decays with the increasing of network loads due to infinite substrate resources. Our joint node-link VNE algorithms, including GA-STP, GA-SEQ and GA-RAN outperformed all competitors in the average acceptance ratio performance. Specifically, GA-STP, GA-SEQ and GA-RAN achieved better average acceptance ratios than MCTS (the best algorithm amongst compared algorithms in general) for more than $\{13.35\%, 14.63\%, 7.5\%\}$ and $\{16.35\%, 14.87\%, 8.65\%\}$ at 40 and 80 Erlangs respectively. They significantly performed better than DPGA, NTANRC-S and D-ViNE for more than 7.4% up to 24.93% at the highest traffic load.

Furthermore, the proposed GA-based algorithms produced higher average revenue and lower embedding cost, which leads to higher R/C as illustrated in Fig. 9a, 9b and 8b respectively. In details, our best joint node-link VNE solution (GA-STP) gained $30.23\%, 34.98\%, 63.23\%$ and 61.56% better average R/Cs than those of aforementioned algorithms at the lowest traffic load. Its overwhelming dominance towards this performance metric was retaining over various arrival rates as depicted in Fig 8b. These results have again confirmed by better average revenue and embedding cost of our approach in Fig. 9a and 9b.

In fact, higher average acceptance and R/Cs are the most desired target for any VN mapping algorithm, and our proposed approaches have successfully proven their great efficiency and significantly reduced the resource fragmentation problem caused by possibly inefficient embedding. With the increasing network loads, node and link utilization was expected to increase as depicted in Fig. 10a and 10b. Specifically, the node utilization of our joint node-link algorithms was remarkably higher at least 13.41% up to 77.83% than all rivals over various traffic loads (Fig. 10a). Our algorithms were indeed capable of accepting more VNRs compared to other heuristics, considering multiple crucial embedding factors during mapping. When the network loads were increasing, joint node-link heuristics still successfully embedded given VNRs by exploiting physical resource capacity more efficiently. In terms of link utilization, GA-STP and GA-SEQ exceedingly leaded this performance metric over the other algorithms thanks to effective path searching approaches as described in Section V-D. Due to the random searching strategy, GA-RAN could not guarantee the link mapping solutions with short path lengths, so GA-RAN did not have an advantage on the link utilization as shown in Fig. 10b. In addition, GA-STP and GA-SEQ performed better average delay than MCTS, the best delay performance among the compared algorithms, for at least 15.33% up to 29.07% as depicted in Fig. 11a.

These dramatic results of our GA-based approaches derive from the fact that the proposed algorithms consider the coordination between VNoM and VLiM at the same time. The novel conciliation mechanism not only handles the remapping strategy intelligently by minimizing the number of virtual nodes and virtual links for remapping, but also reduces the possibility of missing out optimal solutions due to remapping all virtual nodes. Moreover, GA algorithm efficiently explored the search space to approach optimal VNE solutions driven by an efficient multi-objective fitness function. Due to less bandwidth consumed to embed VNRs, abundant residual bandwidth enables to accept more incoming VNRs verified by Fig. 8a. The appealing results of our joint node-link GA-based algorithms are indeed a real challenge for any VNE algorithm we suppose.

2) Amongst Joint node-link algorithms: In this paper, we proposed three joint node-link algorithms according to different path searching methods. They were integrated into GA algorithm for embedding VNRs using the same fitness function to drive the algorithm to optimal VNE solutions.

GA-STP performed best following GA-SEQ and GA-RAN. With low load, GA-SEQ was slightly better than GA-STP for accepting more VNRs but when the load was increasing, GA-STP became efficient in embedding. This could explain why GA-SEQ was little better than GA-STP in node utilization in Fig. 10a. Additionally, GA-STP demonstrated its better VLiM performance through improved embedding cost, average link utilization and average delay by leveraging the shortest path method. Specifically, GA-STP was better than its direct counterpart, GA-SEQ, in the same aforementioned performance metrics for 5.42%, 3.16% and 9.41%, respectively at the arrival rate of 40. Both tended to be asymptotic when the network became more and more congested.

GA-STP, GA-SEQ and GA-RAN are all coordinated approaches. They are different in how the link mapping is done. Comparing with GA-STP and GA-SEQ, GA-RAN is clearly the worst. GA-RAN selects paths randomly, which likely leads to paths with more hops. More hops mean more resource usage at each link that leads to higher link utilization as illustrated in Fig. 10b. This also makes links more congested. GA-RAN tries only several randomly selected paths for each link mapping, which lead to longer path and lower acceptance ratio. In our model, the delay of a path is only dependent on the distance of the path, which is the sum of distances of all the links in the path. Because distances of links are random, in average, longer paths (or paths with longer delay) also mean paths are traversing more hops, they are positively correlated.

In contrast, GA-STP tries the first shortest path, if it fails, we will go through the conciliation process, which also tries the first shortest path for each new node mapping. Therefore, it always leads to shortest paths and lowest utilization. In GA-SEQ, we search feasible link mapping sequentially starting with the shortest first. If a feasible mapping is found, we do not need to go through conciliation process. This will likely lead to longer path and higher utilization than GA-STP. Because it provides more alternative mappings than GA-STP, its acceptance ratio will be higher when load is lower as shown in Fig. 8a. When load is higher, longer paths tend to make congestion worse, which leads to lower acceptance ratio.

Furthermore, we conducted the joint node-link VNE mapping algorithms on different sequential and parallel schemes, and then compared execution time between them to quantify the time reduction towards parallel operation. The distributed parallel paradigm as shown in Fig.3 was deployed for the parallel operation. Average execution time of the proposed algorithms for processing a VNR is shown in Fig. 11b. As such, GA-SEQ was fastest compared to the other algorithms in both experiment schemes. Towards sequential operation, GA-SEQ was faster than GA-STP and GA-RAN for more than 26% and 28% respectively. In this scheme, GA-RAN that is based on random mechanism needed more time to seek for feasible solutions while GA-SEQ due to its simplicity took least time to finish processing a VNR. On the other hand, parallel scheme reveals that shortest path method and random mechanism most likely achieved the same execution time, which means that GA-RAN exploited the distributed machines approaching VNE solutions efficiently with the random manner. GA-SEQ is still a winner since it was faster than the other counterparts for more than 25%. Although, GA-STP performed best within the set of proposed joint node-link VNE algorithms in terms of performance, if we need a more rapid mechanism, an alternative choice could be GA-SEQ. On the convergence of GA, the tendency of fitness values of GA-based algorithms in this paper is depicted in Figure 12 with 95% CIs. Due to an appropriate approach driven by an efficient fitness function, our proposed solutions are rapidly converged after few iterations.

VII. Conclusion

Network virtualization is a primary enabler for the anticipated success of future networks (e.g., virtualized 5G, smart IoT networks); thus, efficient VNE algorithms are eminently desirable. VNE has been well studied in last decade, but there are very few papers dealing with online VNE problem using heuristic or metaheuristic algorithms in a joint manner. In this article, we propose joint node and link embedding approaches relied on GA algorithm for simultaneously solving virtual node and link mappings. When the node mappings are changed, the link mappings are accordingly altered. A heuristic conciliation mechanism is then used to tackle infeasible link mappings due to inappropriate virtual node embedding in GA operations. Furthermore, we deploy our proposed algorithms on a distributed parallel operation scheme in order to reduce time execution and then we make a time comparison of the proposed VNE solutions running on sequential and parallel operations. The extensive evaluation indicates that our joint node-link combination in a single VNE mapping stage based on GA-based approaches outperforms most of state-of-the-art heuristic VNE algorithms in entirely indispensable evaluation metrics we adopted. In fact, current conciliation strategy simply approaches the random mapping instead of attempting all possible combinations, and then selecting the best one among them. Hence, there are some potential to further explore this remapping strategy and we leave this idea for future work.

Acknowledgment

The authors would like to thank anonymous reviewers for their careful reading of our manuscript and many insightful comments and suggestions.

References

[1] A. Hakiri and P. Berthou, "Leveraging SDN for the 5g networks: Trends, prospects and challenges," *CoRR*, vol. abs/1506.02876, 2015. [Online]. Available: http://arxiv.org/abs/1506.02876

[2] I. Ishaq, J. Hoebeke, I. Moerman, and P. Demeester, "Internet of things virtual networks: Bringing network virtualization to resource-constrained devices," in *2012 IEEE International Conference on Green Computing and Communications*, Nov 2012, pp. 293–300.

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

[4] J. Gil Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[5] T. A. Q. Pham, J.-M. Sanner, C. Morin, and Y. Hadjadj-Aoul, "Virtual network function–forwarding graph embedding: A genetic algorithm approach," *International Journal of Communication Systems*, vol. 33, no. 10, p. e4098, 2020, e4098 0.1002/dac.4098. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.4098

[6] B. Addis, G. Carello, and M. Gao, "On a virtual network functions placement and routing problem: Some properties and a comparison of two formulations," *Networks*, vol. 75, no. 2, pp. 158–182, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.21915

[7] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, Feb 2012.

[8] C.-W. Huang, C.-A. Shen, C.-Y. Huang, T.-L. Chin, and S.-H. Shen, "An efficient joint node and link mapping approach based on genetic algorithm for network virtualization," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 2019, pp. 1–5.

[9] K. T. Nguyen, Q. Lu, and C. Huang, "Rethinking virtual link mapping in network virtualization," in *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, 2020, pp. 1–5.

[10] Hong-Kun Zheng, J. Li, Y. Gong, W. Chen, Zhiwen Yu, Z. Zhan, and Ying Lin, "Link mapping-oriented ant colony system for virtual network embedding," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, June 2017, pp. 1223–1230.

[11] G. S. Paschos, M. A. Abdullah, and S. Vassilaras, "Network slicing with splittable flows is hard," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2018, pp. 1788–1793.

[12] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862 – 876, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128609003387

[13] H. Cao, H. Hu, Z. Qu, and L. Yang, "Heuristic solutions of virtual network embedding: A survey," *China Communications*, vol. 15, no. 3, pp. 186–219, March 2018, doi: 10.1109/CC.2018.8332001.

[14] H. Cao, S. Wu, Y. Hu, Y. Liu, and L. Yang, "A survey of embedding algorithm for virtual network embedding," *China Communications*, vol. 16, no. 12, pp. 1–33, Dec. 2019, doi: 10.23919/JCC.2019.12.001.

[15] M. Melo, S. Sargento, U. Killat, A. Timm-Giel, and J. Carapinha, "Optimal virtual network embedding: Node-link formulation," *IEEE Transactions on Network and Service Management*, vol. 10, no. 4, pp. 356–368, 2013, doi: 10.1109/TNSM.2013.092813.130397.

[16] C. Huang and J. Zhu, "Modeling service applications for optimal parallel embedding," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 1067–1079, Oct 2018.

[17] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, “Virtual network embedding through topology-aware node ranking,” *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, p. 38–47, Apr. 2011. [Online]. Available: https://doi.org/10.1145/1971162.1971168

[18] P. Zhang, H. Yao, and Y. Liu, “Virtual network embedding based on the degree and clustering coefficient information,” *IEEE Access*, vol. 4, pp. 8572–8580, 2016.

[19] H. Cao, L. Yang, and H. Zhu, “Novel node-ranking approach and multiple topology attributes-based embedding algorithm for single-domain virtual network embedding,” *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 108–120, Feb 2018.

[20] M. Feng, J. Liao, J. Wang, S. Qing, and Q. Qi, “Topology-aware virtual network embedding based on multiple characteristics,” in *2014 IEEE International Conference on Communications (ICC)*, 2014, pp. 2956–2962.

[21] X. Mi, X. Chang, J. Liu, L. Sun, and B. Xing, “Embedding virtual infrastructure based on genetic algorithm,” in *2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies*, Dec 2012, pp. 239–244.

[22] Z. Zhou, X. Chang, Y. Yang, and L. Li, “Resource-aware virtual network parallel embedding based on genetic algorithm,” in *2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, 2016, pp. 81–86.

[23] P. Zhang, H. Yao, M. Li, and Y. Liu, “Virtual network embedding based on modified genetic algorithm,” *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 481–492, Mar 2019. [Online]. Available: https://doi.org/10.1007/s12083-017-0609-x

[24] L. Boyang, W. Muqing, and Z. Haosen, “Virtual network embedding based on hybrid adaptive genetic algorithm,” in *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, 2019, pp. 1197–1202.

[25] I. Pathak and D. P. Vidyarthi, “A model for virtual network embedding across multiple infrastructure providers using genetic algorithm,” *Science China Information Sciences*, vol. 60, no. 4, p. 040308, Mar 2017. [Online]. Available: https://doi.org/10.1007/s11432-016-9015-3

[26] P. Zhang, X. Pang, G. Kibalya, N. Kumar, S. He, and B. Zhao, “Gcmd: Genetic correlation multi-domain virtual network embedding algorithm,” *IEEE Access*, vol. 9, pp. 67 167–67 175, 2021.

[27] Q. Lu, K. Nguyen, and C. Huang, “Distributed parallel algorithms for online virtual network embedding applications,” *International Journal of Communication Systems*, vol. n/a, no. n/a, p. e4325, e4325 dac.4325. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.4325

[28] S. Haeri and L. Trajković, “Virtual network embedding via monte carlo tree search,” *IEEE Transactions on Cybernetics*, vol. 48, no. 2, pp. 510–521, 2018.

[29] H. Yao, B. Zhang, P. Zhang, S. Wu, C. Jiang, and S. Guo, “Rdam: A reinforcement learning based dynamic attribute matrix representation for virtual network embedding,” *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2018, doi: 10.1109/TETC.2018.2871549.

[30] P. Zhang, C. Wang, G. S. Aujla, and X. Pang, “A node probability-based reinforcement learning framework for virtual network embedding,” in *2020 IEEE 21st International Symposium on ”A World of Wireless, Mobile and Multimedia Networks” (WoWMoM)*, Cork, Ireland, 2020, pp. 421-426, doi: 10.1109/WoWMoM49955.2020.00077.

[31] P. Zhang, C. Wang, C. Jiang, and A. Benslimane, “Security-aware virtual network embedding algorithm based on reinforcement learning,” *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2020, doi: 10.1109/TNSE.2020.2995863.

[32] H. Yao, S. Ma, J. Wang, P. Zhang, C. Jiang, and S. Guo, “A continuous-decision virtual network embedding scheme relying on reinforcement learning,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 864–875, 2020.

[33] P. S. Felipe, M. Vashisht, C. Edoardo, L. Joel, O. S. Kenneth, and C. Jeff, “Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning,” in *arXiv:1712.06567*, 2018, pp. 1–16.

[34] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.

[35] H. Mühlenbein, “Parallel genetic algorithms in combinatorial optimization,” in *Computer Science and Operations Research*, O. BALCI, R. SHARDA, and S. A. ZENIOS, Eds. Amsterdam: Pergamon, 1992, pp. 441 – 453. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780080408064500344

[36] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, “How to model an internetwork,” in *Proceedings of IEEE INFOCOM ’96. Conference on Computer Communications*, vol. 2, March 1996, pp. 594–602 vol.2.

Khoa Nguyen received his M.Sc. degree in Telecommunications Engineering from the University of Sunderland, England, in 2013 and the Ph.D. degree in Electrical and Computer Engineering at the Department of Systems and Computer Engineering, Carleton University, Canada, in 2021, respectively. His main research interests include communication networks, cloud/edge computing, parked vehicle edge computing (PVEC), Internet of Vehicles (IoV), software-defined networks (SDN), network function virtualization (NFV), containerization technologies and machine learning.

Changcheng Huang received his B. Eng. in 1985 and M.Eng. in 1988 both in Electronic Engineering from Tsinghua University, Beijing, China. He received a Ph.D. degree in Electrical Engineering from Carleton University, Ottawa, Canada in 1997. From 1996 to 1998, he worked for Nortel Networks, Ottawa, Canada where he was a systems engineering specialist. He was a systems engineer and network architect in the Optical Networking Group of Tellabs, Illinois, USA during the period of 1998 to 2000. Since July 2000, he has been with the Department of Systems and Computer Engineering at Carleton University, Ottawa, Canada where he is currently a full professor. Dr. Huang won the CFI new opportunity award for building an optical network laboratory in 2001. He is an associate editor of Springer Photonic Network Communications. Dr. Huang is a senior member of IEEE.