Efficient Virtual Network Embedding with Node Ranking and Intelligent Link Mapping

Khoa Nguyen†, Qiao Lu†, and Changcheng Huang†
†Department of Systems and Computer Engineering
Carleton University, Ottawa, ON K1S 5B6, Canada
{*khoatnguyen, qiaolu, huang*}*@sce.carleton.ca*

***Abstract*—Network virtualization (NV) is emerged as a key enabler for the success of the future virtualized networks (e.g. 5G networks and smart Internet of Things (IoT)). Virtual Network Embedding (VNE) that addresses the embedding problems of heterogeneous virtual networks (VNs) onto a physical infrastructure is a main challenge in NV. Network topology attributes and network resource-considered (NTANRC) algorithm is a virtual node mapping mechanism that considers essential network features and global network resources for ranking both substrate and virtual nodes prior to embedding each given virtual network request (VNR). In this paper, we propose NTANRC combined with a distributed parallel Genetic Algorithm (GA) for virtual link mapping, namely NTANRC-GA, to solve online VNE problem. Extensive evaluation results show that our proposed solution not only achieves better performance compared to state-of-the-art VNE algorithms, but also challenges the rapid speed of shortest path (SP) method. NTANRC algorithm and the parallel GA-based algorithm are reverse compliments of each other to achieve an efficient VNE solution.**

***Index Terms*—Network Virtualization, Virtual Network Embedding, 5G-and-beyond networks, smart IoT, Node Ranking, Genetic Algorithm.**

I. Introduction

NV is recognized as a de facto paradigm to make provision for anticipated success of the future networks such as 5G [1], virtualised IoT networks [2]. NV notably allows to share substrate network (SN) resources among multiple VNRs, that enables an isolated coexistence of several VNs on a single physical network. Virtualization technology not only improves network resource utilization of the SN efficiently, but also facilitates the deployment and the evaluation of new architecture designs or network protocols. Subsequently, NV prevents network infrastructure from an avoidable expansion. In VN environment, a service provider (SP) typically converts a requested service/application into a VN, and then conveys to an infrastructure provider (InP) under a VNR. Thereupon, InP will embed the underlying VN onto its physical infrastructure through an optimization process with multiple constraints. Towards InPs, they advocate an efficient resource allocation mechanism that increases their revenue by serving as many VNRs as possible while keeping the embedding costs minimized. A VNR is generally associated with a set of virtual nodes connected via virtual links to form a specific topology, dynamically arriving and residing in the network for a random duration in most scenarios. Embedding VNRs onto the underlying SN with manifold topology and rigid resource requirements is known as a VNE problem.

VNE process can be divided into two sub-problems: Virtual Node Mapping (VNoM) and Virtual Link Mapping (VLiM). VNE has been proven to be $\mathcal{NP}$-Hard either for VNoM or VLiM [3]. The common formulated optimization models (e.g. Integer Linear Programming (ILP)) are usually recommended to gain optimal VNE solutions, but they face several issues of scalability, complex implementation, and high time consumption. These solutions cannot be indeed tailored for online VNE problems. Indeed, most of research papers have adopted light-weighted heuristic algorithms to deal with the aforementioned obstacles of the optimization models. Several heuristic algorithms [3]–[8] have been proposed over the past decade, embedding VNRs in separate mapping stages. These algorithms relaxed the integer constraints to achieve feasible VNE in polynomial time [3] or considered a single topology attribute and local network resources [4]–[6] for node mapping stage, or exploited simplified global network resources for VNoM problems [7], [8]. NTANRC in [9] adopted an efficient node-ranking approach based on five topology attributes (e.g. node degree, node strength, node distance, farness/closeness and link interference) and global network resources (e.g. node location, node capacity, link bandwidth and link propagation delay) to rank all substrate and virtual nodes before embedding each VNR. This node-ranking approach coordinated node and link constraints to provide a better performance of embedding VNRs.

In fact, a vast number of research papers including [3]–[11] merely focused on approaching an efficient node mapping, and completely put trust in k-shortest path or multicommodity flow (MCF) algorithms for virtual link mapping. Unlike previous papers, [12] primarily centralized VLiM stage utilizing a novel distributed parallel GA-based algorithm, namely IDPA, with a very simple node mapping based on a greedy method. Through its evaluation, this paper proved that VLiM also plays a crucial role in attaining an efficient VNE algorithm along with the inevitably important role of VNiM. This intelligent VNE algorithm outperformed its competitors not only in performance, but also in speed due to a complicated link mapping algorithm implemented in a proper embedding scheme.

In this paper, we propose the NTANRC-GA algorithm, a dramatic combination between an efficient node-ranking approach and an intelligent GA-based algorithm in aimed at obtaining an effective VNE algorithm. Our proposed algorithm, that exploits a set of distributed parallel machines, enables to embed multiple link mapping requests at the same time so as to reduce the execution time. We mainly aim to increase profit of an InP by maximizing the VNR acceptance ratios while minimizing the embedding costs. To the best of our knowledge, this is the first paper that efficiently deploys a node-ranking approach incorporated with a complicated link mapping solution to solve VNE problems. The results show that our proposed VNE solution improves the acceptance ratios of VNRs, revenue to cost ratios and link resource utilization compared to NTANRC [9], IDPA [12] and three VINE algorithms [3].

The remainder of this paper is organized as follows: the network model is formulated in Section II. NTANRC approach for VNE node mapping and intelligent parallel GA-based algorithm are described in Section III and Section III-B, respectively. Performance evaluation is introduced in Section IV while related work is presented in Section V. Finally, Section VI is a conclusion of this paper.

II. Network Model and Problem Descriptions

A. Virtual Network Assignment

SN is modelled as a weighted undirected graph $G^s = (N^s, L^s)$, where N^s is the set of substrate nodes and L^s is the set of substrate links. A substrate node $n^s \in N^s$ with a geographical location $loc(n^s)$ has the available CPU capacity $c(n^s)$, whereas each physical link $l^s \in L^s$ between any two physical nodes possesses a $b(l^s)$ bandwidth capacity. For simplification, memory and storage resources are excluded in this paper. Let model the i^{th} arriving VNR 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 while L_i^v is the set of virtual links. Each virtual node $n_i^v \in N_i^v$ has a requested CPU capacity $c(n_i^v)$, whereas a virtual edge $l_i^v(s_i^v, d_i^v) \in L_i^v$ between a virtual source node s_i^v and a virtual destination node d_i^v possesses a required bandwidth capacity $b(l_i^v)$. Each VNR normally prefers a mapping radius $D(n_i^v)$ revealing how far virtual node n_i^v can be allocated from $loc(n_i^v)$. Mapping the i^{th} VNR G_i^v onto the SN G^s can be decomposed into

978-1-7281-9486-8/20/$31.00 ©2020 IEEE

two main components as determined above: VNoM and VLiM. In node mapping stage, a virtual node of a VNR can be embedded onto a substrate node $\mathcal{A}_{\mathcal{N}} : N_i^v \rightarrow N^s$, with $n^v \in N_i^v$ subject to:

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

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

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

$$R_N(n^s) = c(n^s) - \sum_{n^v \rightarrow n^s} c(n_i^v) \tag{4}$$

where $n^v \rightarrow n^s$ defines the virtual node n^v embedded onto physical node n^s, and the distance between i^s and j^d is expressed by $\mathcal{D}(i^s, j^d)$ whereas $R_N(n^s)$ denotes the remaining CPU capacity of a substrate node. Indeed, each virtual link can be commonly mapped onto the corresponding physical path including one or more substrate links. The unsplittable link mapping can be expressed as $\mathcal{A}_{\mathcal{L}} : L_i^v \rightarrow L^s$ while $l_i^v = (s_i^v, d_i^v) \in L_i^v$, $\mathcal{E}^s(\mathcal{A}_{\mathcal{L}}(l_i^v))$ denotes a set of all available physical paths from the source $\mathcal{A}_{\mathcal{N}}(s_i^v)$ to destination node $\mathcal{A}_{\mathcal{L}}(d_i^v)$.

$$\mathcal{A}_{\mathcal{L}}(s_i^v, d_i^v) \subseteq \mathcal{E}^s(\mathcal{A}_{\mathcal{N}}(s_i^v), \mathcal{A}_{\mathcal{N}}(d_i^v)) \tag{5}$$

subject to:

$$R_L(e^s) \geq b(l_i^v), \forall e^s \in \mathcal{E}^s(\mathcal{A}_{\mathcal{L}}(l_i^v)) \tag{6}$$

$$R_L(e^s) = \min_{l^s \in e^s} R_L(l^s) \tag{7}$$

$$R_L(l^s) = b(l^s) - \sum_{l_i^v \rightarrow l^s} b(l_i^v) \tag{8}$$

where $R_L(e^s)$ is the available bandwidth of a substrate path $e^s \in \mathcal{E}^s$, and $R_L(l^s)$ is the residual substrate link capacity.

B. Performance metrics

The major objective of VNE algorithm is to maximize the InP's revenues while minimizing the embedding cost. Thus, the ratio between revenue over cost should be considered to estimate the performance of VNE algorithms, adhering to the acceptance ratio. For instance, high acceptance ratio while the average revenue to cost ratio is low is unfavourable since this result reveals that the substrate resources are underutilized. In this article, the InP revenue is computed as the sum of total virtual resources mapped on the SN over time while the cost of the i^{th} VNE $C(G_i^v)$ is the sum of total network resources allocated to the i^{th} VN. Subsequently, the revenue to cost ratio of i^{th} VNR G_i^v can be expressed as below:

$$\Upsilon(G_i^v) = \frac{\mathcal{R}(G_i^v)}{C(G_i^v)} = \frac{w_b * \sum_{l_i^v \in L_i^v} b(l_i^v) + w_n * \sum_{n_i^v \in N_i^v} c(n_i^v)}{\sum_{n_i^v \in N_i^v} c(n_i^v) + \sum_{l_i^v \in L_i^v} \sum_{l^s \in L^s} f_{l^s}^{l_i^v}} \tag{9}$$

where $\mathcal{R}$ and C are the generated revenue and network cost respectively. $b(l_i^v)$ and $c(n_i^v)$ are the requested bandwidth of the virtual link l_i^v and the requested CPU of the virtual node n_i^v while w_b and w_n $f_{l^s}^{l_i^v}$ defines the bandwidth of substrate link l^s that is allocated to the virtual link l_i^v.

Acceptance ratio: is expressed as the ratio between the number of accepted VNRs over the number of arrived VNRs during the interval time τ is calculated as following:

$$\mathcal{A}_c^\tau = \left| \frac{\xi^a(\tau)}{\xi(\tau)} \right| \tag{10}$$

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

Remaining bandwidth: the residual bandwidth of a SN can be calculated as following:

$$\mathcal{R}_m(L^s) = \sum_{l^s \in L^s} (b(l^s) - \sum_{l_i^v \rightarrow l^s} b(l_i^v)) \tag{11}$$

When new VNRs arrived, InP calculates the remaining network resources, and then attempts to map such VNRs onto the SN depending on the residual network resource information obtained. Higher remaining bandwidth leads to greater opportunity for accepting the upcoming VNs.

Fitness Function (FF): determine which link solutions will reproduce and remain in next generation, relevant to the predefined objectives to be optimized in our proposed GA-based algorithm in Section III-B. Consequently, fitness function is used to examine the quality of each VLiM solution among several feasible ones. Its fitness values are able to present a scientific proof of selecting the corresponding solutions in GA operators. Similar to [12], we take hop-count and bandwidth into consideration in fitness function. Less hop-count solution is preferable since it is substantially associated with less bandwidth consumption, leaving more residual network bandwidth that increases the possibility of the upcoming VNRs being accepted.

Fitness function $\mathcal{F}(\mathcal{S}_z)$ is eventually calculated as below:

$$\mathcal{F}(\mathcal{S}) = \sum_{l_i^v \in L_i^v} \left(\frac{b(l_i^v) * w_\alpha}{b(l_i^v) + R_L(\mathcal{A}_{\mathcal{L}}(l_i^v))} + \frac{1}{h_{\mathcal{A}_{\mathcal{L}}(l_i^v)}} * w_\beta \right) \tag{12}$$

where, $\mathcal{S}$ and h are a feasible solution and hop-count of the link mapping solution of l_i^v respectively. w_α and w_β are weight parameters equivalent to bandwidth and hop-count factors.

III. NTANRC-GA Algorithm

NTANRC-GA implements NTANRC algorithm [9] for addressing VNVoM while deploying the distributed parallel GA algorithm [12] for link mapping. It means that the output of NTANRC algorithm becomes the input for VLiM. NTANRC algorithm efficiently seeks the potential embedding nodes for VNRs. Besides, the novel GA-based algorithm that is based on a distributed parallel scheme efficiently reduces the embedding cost, execution time, and improves link utilization.

A. NTANRC Algorithm for Node Mapping

NTANRC collects five network topology attributes for its node-ranking adoption including node degree, node strength, distance, farness and closeness and finally link interference. As shown in [9], there are two sub-approaches (S: stable and D: direct), but we prefer S-approach due to its better performance.

Node degree: Let $\tilde{\mathcal{T}}_d(.)$ denote the function that counts the total number of adjacent links of node n^s, degree of a substrate node is determined as follows:

$$\Gamma(n^s) = \tilde{\mathcal{T}}_d(n^s) \tag{13}$$

Node strength is defined by the function $\tilde{\mathcal{T}}_s(.)$ that counts the sum of all adjacent link bandwidth of the substrate node n^s.

$$\Theta(n^s) = \tilde{\mathcal{T}}_s(n^s) \tag{14}$$

Farness and Closeness of a node: Farness of a node n^s is the sum of its shortest distances to all the other possible nodes whereas Closeness is reciprocal of the Farness as defined in (15) and (16).

$$F(n^s) = \sum_{m^s \in N^s} \mathcal{D}(n^s, m^s) \tag{15}$$

$$Z(n^s) = \frac{1}{F(n^s)} \tag{16}$$

Link Interference: Interference of a physical link $l_{(n^s,m^s)}^s \in L^s$ between node n^s and node m^s describes its contribution to the network connectivity, relying on the idea that traffic routed in a path can be minimized traffic interference in the future.

$$I(l_{(n^s,m^s)}^s) = \frac{F(n^s)}{\Gamma(n^s)} + \frac{F(m^s)}{\Gamma(m^s)} \tag{17}$$

In node-ranking approach, topology attributes and the global network resources are simultaneously quantified in which the node capacity, node location, link bandwidth and link propagation delay are considered as global resource parameters. Accordingly, interaction between two nodes n^s and m^s denoted as $\Phi_{(n^s,m^s)}$ is defined as follows:

$$\Phi_{(n^s,m^s)} = \epsilon_I \frac{\mathcal{B}(n^s)\mathcal{B}(m^s)}{\mathcal{D}(n^s, m^s)^2 d_{\mathbb{P}}(n^s, m^s)^2} \tag{18}$$

$$\Phi_{(n^s)} = \sum_{n^s \neq m^s, m^s \in N^s} \Phi_{(n^s, m^s)} \tag{19}$$

where ϵ_I is a constant, and Eq. 19 expresses the interaction of a node n^s with remaining nodes in the entire substrate network. Moreover, $\mathcal{B}$ denotes the resource block of a substrate node that can be defined as follows:

$$\mathcal{B}(n^s) = c(n^s)\Theta(n^s) \sum_{m^s \neq n^s, m^s \in N^s} I(l^s_{(n^s, m^s)}) \tag{20}$$

Normalized $\mathcal{B}(n^s)$ and $\Phi_{(n^s)}$ towards substrate node n^s are formulated as below:

$$\bar{\mathcal{B}}(n^s) = \frac{\mathcal{B}(n^s)}{\sqrt{\sum_{n^s \in G^s} \mathcal{B}(n^s)^2}} \tag{21}$$

$$\bar{\Phi}_{(n^s)} = \frac{\Phi_{(n^s)}}{\sqrt{\sum_{n^s \in G^s} \Phi_{(n^s)}{}^2}} \tag{22}$$

By calculating all percentage values $\bar{\Phi}_{(n^s)}$ of the substrate network G^s, an initial node-ranking vector T_0 has been made, where $T_0 = (\bar{\Phi}_{(n^0)}, \bar{\Phi}_{(n^1)} \cdots \bar{\Phi}_{(n^{|N|})})^T$. For each node $n^s \in N^s$, its node-ranking value can be calculated as follows:

$$\nu_{n^s} = (1-\rho)\bar{\mathcal{B}}(n^s) + \rho \sum_{n \neq m, m \in N(n^s)} \Phi_{(n^s)} \nu_{m^s} \tag{23}$$

where ρ is the damping factor and $N(n^s)$ is the set of all nodes that have a path with the node n^s in the substrate network. As a result, a vector $\mathcal{V}$ formed by the node-ranking values of all substrate nodes can be shown as below:

$$\mathcal{V}_{n^s} = (1-\rho)\bar{\mathcal{B}}(n^s) + \rho \mathcal{M} \mathcal{V}_{n^s} \tag{24}$$

where $\mathcal{V}_{n^s} = (\nu_{n^0}, \nu_{n^1}, \cdots, \nu_{n^{|N|}})^T$, and $\bar{\mathcal{B}}(n^s) = (\bar{b}(n^0), \bar{b}(n^1), \cdots, \bar{b}(n^{|N|}))^T$. $\mathcal{M}$ is the transition matrix ($|N| \times |N|$) where each element is calculated based on (21) and (22). Details on a stable node-ranking approach are described in Algorithm 1 that is utilized to calculate the node-ranking values. NTANRC requires that all nodes including substrate and virtual nodes should be ranked and sorted following a decreasing order of the node-ranking values. Similar to substrate nodes, the node ranking values for virtual nodes are computed in the same procedures. When a VNR arrives, virtual nodes are calculated node-ranking values and then the virtual node with highest node-ranking value is first mapped to the substrate node that achieved highest node-ranking value, meeting node location and capacity constraints. In a VNR, virtual nodes cannot share the same substrate node. The mapping is repeated until all virtual nodes are successfully mapped to corresponding substrate nodes. The output of NTANRC algorithm as node mapping results is then utilized as the input to the parallel GA-based algorithm [12] to deal with virtual link mapping.

Algorithm 1 Stable Node-Ranking Approach

Input: a network G = (N,L), δ: small positive number
Output: a ranking vector $\mathcal{V}$ corresponding to a given G
Calculate matrix $\mathcal{M}$ and initial vector $\mathcal{V}_0(T_0)$
Define iteration number k and variable w.
while $w \geq \delta$ **do**
  $\mathcal{V}_{k+1}$ = (1- ρ)$\bar{\mathcal{B}}(n^s)$ + ρ $\mathcal{M}$ $\mathcal{V}_k$;
  w = $\|\mathcal{V}_{k+1} - \mathcal{V}_k\|$;
  k = k +1;
end while
$\mathcal{V}_k$ = $\mathcal{V}_{k+1}$

B. Parallel GA-based Algorithm for Link Mapping

Distributed and Parallel computing has considered as an efficient mechanism to address large and complex problems with lower costs and less time consumption by concurrency support. GA is an appealing AI approach for solving constrained or unconstrained optimization problems. A conventional GA usually includes four main operators: initialization, selection, crossover and mutation. An intelligent GA-based orchestration algorithm for virtual link mapping stage not only approached an efficient VNE solution, but also accelerated the embedding speed due to a distributed parallel paradigm [12]. A proposed parallel GA scheme is presented in Fig 1. Each working machine independently runs GA algorithm with a pre-defined number of iterations to explore feasible solutions for virtual link mapping. The best-matching VLiM outcome is selected among these parallel machines. All virtual link requests (VLRs) within a VNR are embeded at once. Accordingly, a chromosome $\mathcal{C}_f$ that consists of several genes g^j_i denotes a feasible link mapping solution for all VLRs. Each gene g^j_i is associated with a substrate path in which i and j indicate the current chromosome and virtual link respectively.

1) Initial path pool generation To serve link embedding procedures, a path database for mapping VLRs is deliberately created since we know that our SN is static. During its path generation, a k-shortest path algorithm e.g. Dijkstra's algorithm is deployed to find all possible k-shortest paths for each pair of source-destination.

2) Working node:

Population Initialization: Each working machine normally begins with a population initialization step. There are M chromosomes and each has N genes. An initial population $\mathcal{P}$ (MxN size) at k^{th} working machine is represented as follows:

$$\mathcal{P} = \begin{bmatrix} \mathcal{C}_1 \\ \mathcal{C}_2 \\ \vdots \\ \mathcal{C}_f \\ \vdots \\ \mathcal{C}_M \end{bmatrix} = \begin{bmatrix} g^1_1 & \cdots & g^j_1 & \cdots & g^N_1 \\ g^1_2 & \cdots & g^j_2 & \cdots & g^N_2 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ g^1_f & \cdots & g^j_f & \cdots & g^N_f \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ g^1_M & \cdots & g^j_M & \cdots & g^N_M \end{bmatrix} \tag{25}$$

Each gene that is associated with a potential link mapping solution is randomly selected from the initial path pool. Such solution must pass a feasibility check to be a potential embedding solution. N genes already passed the feasibility check compose a chromosome that is acknowledged as a feasible solution for the corresponding VLiM.

Selection: A fitness-based proportionate selection scheme which is based on a cumulative sum of the fitness relative weights (12) is utilized to select parents from the initial population.

$$\mathcal{P} = \begin{bmatrix} \mathcal{C}_1 \\ \vdots \\ \mathcal{C}_s \\ \vdots \\ \mathcal{C}_r \\ \vdots \\ \mathcal{C}_M \\ \mathcal{C}_{M+1} \\ \mathcal{C}_{M+2} \end{bmatrix} = \begin{bmatrix} g^1_1 & \cdots & g^{j^c}_1 & g^{j^c+1}_1 & \cdots & g^N_1 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ g^1_s & \cdots & g^{j^c}_s & g^{j^c+1}_s & \cdots & g^N_s \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ g^1_r & \cdots & g^{j^c}_r & g^{j^c+1}_r & \cdots & g^N_r \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ g^1_M & \cdots & g^{j^c}_M & g^{j^c+1}_M & \cdots & g^N_M \\ g^1_s & \cdots & g^{j^c}_s & g^{j^c+1}_r & \cdots & g^N_r \\ g^1_r & \cdots & g^{j^c}_r & g^{j^c+1}_s & \cdots & g^N_s \end{bmatrix} \tag{26}$$

Crossover: This procedure combines parental chromosomes to generate a new offspring for next generation. $\mathcal{C}_s$ and $\mathcal{C}_r$ denote two parental chromosomes with their indexes s and r in initial population, whereas new descendant chromosomes are described as $\mathcal{C}_{(M+1)}$ and $\mathcal{C}_{(M+2)}$ respectively. j^c indicates a random crossover point between any genes within N length, which is randomly chosen. Offspring is established by swapping parental genes starting from the crossover point $j^c + 1$ to the end as illustrated in 26.

Mutation: This operator implements a random change to an individual parent to produce a new offspring. A mutation point denoted as j^m is randomly generated. At this point, a new gene chosen from the path database replaces an existing one within the in-processed chromosome to form a new child. j^m is a random mutation point, and $g^{j^m}_{r'}$ denotes a new gene that replaced the existing one in $\mathcal{C}_{(M+1)}$. The new mutation solution $\mathcal{C}'_{(M+1)}$ after replacement can be represented as $\mathcal{C}'_{(M+1)} = [g^1_s \cdots g^{j^m}_{r'} \cdots g^N_s]$.

3) Solution Sorting and Terminations Sorting procedure selects

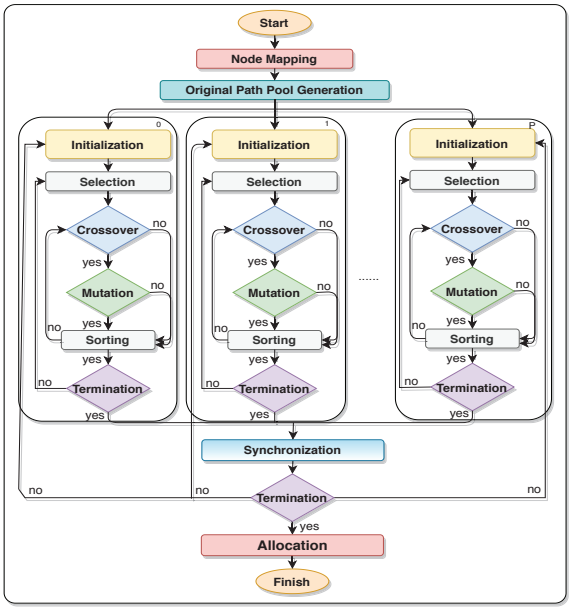


Fig. 1: *Parallel operation scheme*

the best embedding solution among the feasible ones based on their fitness values, and then it is conveyed to synchronization step for a global ranking. A parallel operation generally adopts a series of concurrent processes, and each accomplishes its particular job in different time. However, waiting for the last process finishing its assigned task is frustrating, and this attempt might not guarantee a desired outcome. Thus, the master node will terminate GA algorithms at worker nodes if the current best VLiM solution has not been consecutively improved for t times, where t denotes a termination parameter.

4) Synchronization and VNR allocation

This operation is deliberately to determine the final VNE solution for the corresponding VLiM request by a global ranking process. This final selection is relied on highest achieved FF values compared with those of all feasible embedding solutions conveyed from the working nodes. As a result, the VNR will be accepted and then allocated onto SN based on the information of the virtual node and link mapping solutions found. The residual network resources will be consequently updated in advance.

IV. Performance Evaluation

We compare our proposed NTANRC-GA algorithm with several competitors including IDPA [12], NTANRC-S [9], D-ViNE, R-ViNE, and G-SP [3]. These algorithms are evaluated in various performance metrics including acceptance ratio, average revenue to cost ratio, remaining bandwidth and execution time.

A. Simulation setup

We deployed a discrete-event simulator to assess our proposed VNE solution with parameters similar to those in [3]. Hence, a popular GT-ITM topology generator [13] has been utilized to generate SN and VNs. SNs are configured with average 50 nodes, that are randomly placed on a 25×25 Cartesian plane. These are randomly connected to average 140 edges adopting Waxman model with $\alpha = 0.5$ and $\beta = 0.2$, where α indicates the maximal edge probability and β determines the edge length. CPU and bandwidth capacity of SNs are uniformly generated between 50 and 100 units, whereas VNRs dynamically arrive following the Poisson process with an average rate λ varying from 4 to 8 VNs per 100 time units. Lifetime of VNRs follows an exponential distribution with an average value of $\mu = 1000$ time units. Hence, the load of VNRs can be quantified by $\frac{\lambda}{\mu}$ Erlangs. Additionally, the number of virtual nodes in each VNR is uniformly distributed between 2 and 10. CPU capacity and bandwidth requirements of VNRs are uniformly distributed between 0 to 20 and 0 to 50 respectively. Similar to [14], we set $w_b = w_n = 1$ in this paper. Each simulation run for $50,000$ time units, 50 times longer than the average lifetime of a VN to exceptionally generate a large number of independent samples. All performance figures were based upon average values with 95% confidence interval. The error bars were very small due to a large number of samples used, which proved that our simulation results were obviously reliable. For better presentation, we plotted figures with different colors and markers.

B. Evaluation Results

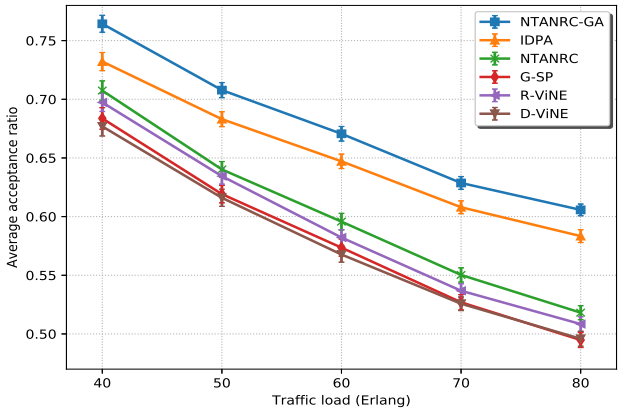
Simulation results are preferably shown in Figure 2 and 3. NTANRC-GA achieved highest acceptance ratio by accepting more VNRs with significantly less costs than all rivals, which resulted in much higher average revenue to cost ratios as depicted in Fig 2. Achieving higher both acceptance and revenue to cost ratios simultaneously is desirable for any VNE algorithm. Our proposed solution proved that it could be feasible. For example, NTANRC-GA improved the acceptance ratios of IDPA as well as NTANRC, and was better than R-ViNE (the best performance algorithm in [3]) up to $2.24\%, 8.76\%$ and 9.73% at 80 Erlang (Fig. 2a) respectively. Our proposed VNE solution gained $9.07\%, 14.08\%$ and 16.60% better revenue to cost ratios than those of aforementioned algorithms at the same traffic load as illustrated in Fig 2b. In this paper, average remaining bandwidth of all compared algorithms was selected and illustrated in Fig. 3a. With the intelligent link mapping, our VNE solution consumed less bandwidth to embed given link mapping requests due to the efficient FF as shown in Fig. 3a. Moreover, we compared the execution time for embedding a VNR among algorithms where those of NTANRC-GA and IDPA algorithms are almost similar because they use the same link mapping mechanism. As depicted in Fig .3b, NTANRC-GA is 36.26% and 40.11% absolute faster than NTANRC and G-SP algorithms, respectively. The reason is that both NTANRC-GA and IDPA deploy the distributed parallel GA-based algorithm for link mapping stage instead of the SP method used in NTANRC and G-SP, which was proved to be considerably faster than SP in [12]. The remarkable performance came from the effective node-ranking method, that simultaneously took network topology attributes and global network resources into account, and the intelligent GA-based approach exploring the search space efficiently to achieve more feasible link mapping solutions with lower time consumption due to a proper parallel operation paradigm.

V. Related Work

Consolidating research endeavors in NV, the authors in [14] contributes a comprehensive survey to this research field. VNE problem is $\mathcal{NP}$-hard in nature, that is intractable to be solved with Integer Programming (LP). Thus, most VNE papers are concentrated on seeking for efficient heuristic algorithms due to the computational complexity of extract methods. [3] introduced a coordinated node and link approach for virtual node embedding by relaxing the intractable integer constraints, and then using rounding techniques to choose unique node mapping. Huang et al. in [15] mainly extended [3] with a novel node splitting scheme and node collocation. The authors in [4] proposed a topology-aware node embedding deploying the Markov Random Walk model to quantify node capacity and its joint link bandwidth. The paper [5] explored seven different topological attributes for better coordinate node and link mapping and proposed various node-ranking approaches. Zhang et al. [6] took the node degree and clustering coefficient information to enhance the metric of node importance which was adopted to rank the substrate nodes, aiming to determine the nodes with the highest potential for embedding VNRs. Gong et al. [7], [8] proposed node-ranking approaches that only considered the simplified global network resources for the node mapping stage without essential topology attributes. Inspired from Google PageRank algorithm, [9] proposed an efficient node-ranking algorithm to rank both virtual and substrate nodes prior to embedding each VNR, which is based on five topology attributes and global network resources. Nguyen et al. [12] proposed a distributed parallel GA-based algorithm for the link mapping phase, which confirmed the critical role of VLiM for approaching an efficient VNE solution. Recently, reinforcement learning algorithms have been proposed to solve VNE problems and optimize their performance [10], [11].

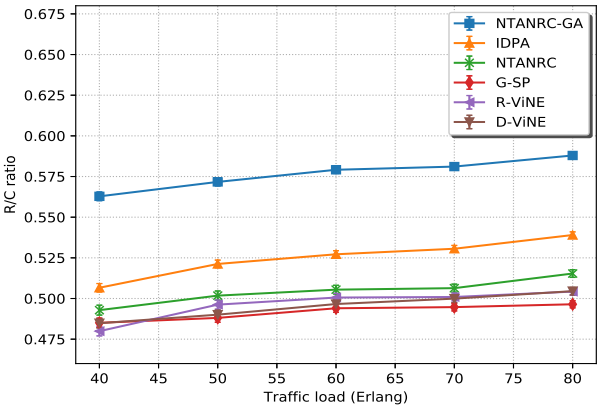
VI. Conclusion

NV is indisputably a major factor of the anticipated success of future network architectures (e.g. 5G, virtualized IoT networks) so that an efficient VNE algorithm is highly desirable. In this paper, we propose the node-ranking approach that is based on network



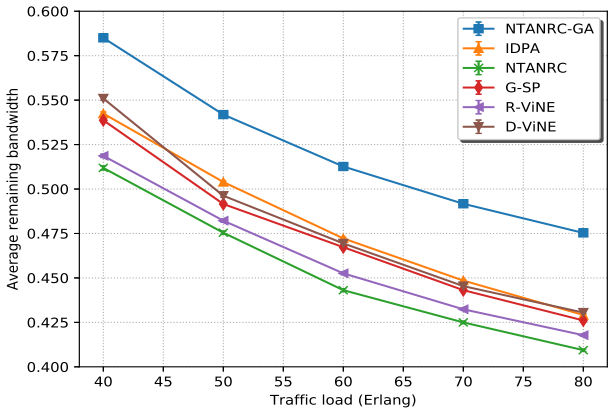
(a)

Fig. 2: **(a)** VNR Acceptance Ratio



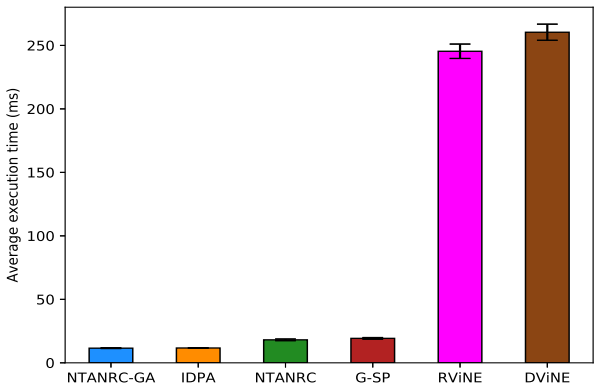
(b)

(b) Average revenue to cost ratio



(a)

Fig. 3: **(a)** Average remaining bandwidth



(b)

(b) Average CPU execution time

topology attributes and global network resources for virtual node mapping stage and the intelligent parallel GA-based algorithm for link mapping stage to solve online VNE problem. Our proposed NTANRC-GA algorithm outperforms state-of-the-art algorithms in all matrices towards performance and operation time. We can conclude that NTANRC and intelligent GA-based algorithms are reverse compliments of each other to achieve an efficient VNE solution. In future work, we will investigate simultaneously embedding nodes and links in one-stage mapping utilizing Genetic Algorithm.

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] 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.

[4] 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

[5] 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.

[6] 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.

[7] Long Gong, Yonggang Wen, Zuqing Zhu, and T. Lee, "Revenue-driven virtual network embedding based on global resource information," in *2013 IEEE Global Communications Conference (GLOBECOM)*, 2013, pp. 2294–2299.

[8] L. Gong, H. Jiang, Y. Wang, and Z. Zhu, "Novel location-constrained virtual network embedding lc-vne algorithms towards integrated node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 24, no. 6, pp. 3648–3661, 2016.

[9] 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.

[10] 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.

[11] Z. Yan, J. Ge, Y. Wu, L. Li, and T. Li, "Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1040–1057, 2020.

[12] K. T. D. Nguyen and C. Huang, "An intelligent parallel algorithm for online virtual network embedding," in *2019 International Conference on Computer, Information and Telecommunication Systems (CITS)*, Aug 2019, pp. 1–5.

[13] 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.

[14] 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

[15] 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.