Joint Node-Link Embedding Algorithm based on Genetic Algorithm in Virtualization Environment

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

***Abstract*—Virtual network embedding (VNE), that efficiently tackles the mapping problems of heterogeneous virtual networks onto a shared physical infrastructure meeting rigid resource constraints, is the major challenge in network virtualization (NV). VNE is widely known as $\mathcal{NP}$-hard due to its intractable computation. The majority of VNE solutions have concentrated upon virtual node mapping (VNoM) and virtual link mapping (VLiM) separately. Uncoordinated approaches would facilitate algorithmic implementation, but they lead to low acceptance ratio, network revenues and high embedding cost. In this paper, we propose a new approach relied on Genetic Algorithm (GA), that coordinately joints node and link mappings where the link embedding is based on a fast and efficient sequential path searching method. A novel heuristic conciliation algorithm is presented to deal with a set of infeasible link mappings during producing VNE solutions in GA's operations. Extensive evaluation results indicate that our proposed approach outperforms state-of-the-art VNE algorithms in all adopted performance metrics.**

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

I. INTRODUCTION

NV is widely envisioned as a promising paradigm for the foreseen success of future networks such as virtualized fifth generation [1], smart Internet of Things (IoT) networks [2]. Virtualization efficiently enhances network utilization by sharing physical resources among several VNRs and provides an isolated coexistence between VNs on a shared substrate network (SN). A service provider (SP) in virtualization environment typically converts a service/application into a VN, and then transfers it to the corresponding infrastructure provider (InP) under a form of a VNR. The InP is responsible for embedding the VN on SN, satisfying multiple stringent resource constraints. VNE, that is one of the most challenging tasks in NV, allows to embed multiple VNRs on the shared SN with various topology and rigid resource demands. VNE is commonly acknowledged as a generalization of second stage of network function virtualization resource allocation (NFV-RA), or virtual network function forwarding graph embedding (NFV-FGE) due to a mutual problem domain. The main goal of both NFV-RA and VNE is allocating VNRs on the top of physical infrastructure efficiently [3]. In addition, NFV-FGE is the most critical problem of NFV-RA following virtual network functions (VNFs) chain composition (or service function chaining) and VNFs scheduling, where the chain composition is normally neglected in most research papers [3]. In some particular cases and topology, VNE is even revealed to be more complicated than NFV-FGE [4], [5]. VNE is $\mathcal{NP}$-Hard either for VNoM or VLiM [3], [6]. Thus, the majority of research papers in this filed have merely focused on designing efficient heuristics or meta-heuristic algorithms to address the hurdles of the exact optimization models.

In similarity to NFV-FGE, the VNE process includes two different stages: VNoM and VLiM. Most of VNE approaches have been solved these mapping stages separately. Specifically, a large number of VNE algorithms are proposed to deal with the VNoM problem, leaving the VLiM phase for the popular shortest path methods (e.g., Dijkstra's algorithm), multi-commodity flow (MCF) or recently the distributed parallel GA-based algorithm [7]. Decoupling can simplify the complex implementation of VNE algorithms, but the common solution has apparently to sacrifice some degrees of optimality. Furthermore, a lack of coordination between VNoM and VLim phases could result in low acceptance ratio and low network revenue accordingly. For example, VNoM stage can determine potential substrate nodes for embedding virtual nodes of a given VNR, but there might have a situation that no feasible path is found for virtual link that connects a pair of the already-mapped virtual nodes. In that case, the corresponding VNR is definitely rejected. It is also observed that the most common failures of VNE problem invariably derive from the VLiM stage [8]. In this paper, we propose an efficient VNE algorithm relied on GA algorithm to address the VNE problem by jointly embedding virtual nodes and corresponding virtual links. GA tries to seek for potential solutions for all virtual nodes of the VNR. After that, the corresponding virtual links will be consecutively mapped by a sequential path searching method. There might have no feasible paths found for virtual link requests due to network congestion, the current virtual nodes embedding should be reexamined with a goal of minimizing node and link mappings revisited. Consequently, we present a novel conciliation algorithm to handle this issue. Additionally, we propose a distributed parallel GA-based paradigm exploiting a set of distributed parallel machines in order to reduce the operation time. An execution time comparison between sequential and parallel manners is also conducted. In this work, we only consider unsplittable-enabled mapping solutions for VNE problem.

The remainder of this paper is organized as follows: the network model is formulated in Section II. Genetic Algorithm approach for jointly node-link VNE mapping is described in Section III. Performance evaluation is introduced in Section IV.Finally, Section V is a conclusion of this paper.

II. NETWORK MODEL AND PROBLEM DESCRIPTIONS

A. *Virtual Network Assignment*

Similar to [6], our SN is modelled as a weighted undirected graph $G^s = (N^s, L^s)$, in which N^s is a set of physical nodes and L^s is a set of physical links. A substrate node $n^s \in N^s$ has its geographical location $loc(n^s)$ and the available CPU capacity $c\,(n^s)$, whereas a substrate link $l^s \in L^s$ between two substrate nodes has a $b\left(l^s\right)$ bandwidth capacity. Memory and storage resources are neglected in this paper for simplification. Similarly, the i^{th} VNR can be modelled as a weighted undirected graph denoted as $G_i^v = (N_i^v, L_i^v)$, where N_i^v and L_i^v are the sets of virtual nodes and virtual links respectively. Every virtual node $n_i^v \in N_i^v$ requires a 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 demands a bandwidth capacity $b\left(l_i^v\right)$. Each VNR prefers an embedding radius $D(n_i^v)$ that exposes how far n_i^v is allocated from $loc(n_i^v)$.

This paper is aimed at multiple objectives consisting of maximizing average acceptance ratio, average revenue, reducing lower cost and improving node and link utilization meeting several node and link constraints:

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

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

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

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

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

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

$$R_L(l^s) = b(l^s) - \sum_{l_i^v \rightarrow l^s} b(l_i^v) \tag{7}$$

where $\mathcal{A}_{\mathcal{N}}(n_i^v)$ denotes the mapping solution of the virtual node. $\mathcal{D}(i^s, j^d)$ represents the distance between i^s and j^d, and $R_N(n^s)$ is the residual CPU capacity of the physical node. Moreover, $\mathcal{E}^s(\mathcal{A}_{\mathcal{L}}(l_i^v))$ is a set of all available physical paths from the source $\mathcal{A}_{\mathcal{N}}(s_i^v)$ to destination node $\mathcal{A}_{\mathcal{N}}(d_i^v)$. $R_L(e^s)$ and $R_L(l^s)$ denote available bandwidth of a substrate path $e^s \in \mathcal{E}^s$ and the remaining physical link capacity respectively.

An embedding solution is defined as "feasible" if and only if it satisfies all resource constraints (1)-(4) for VNoM and (5)-(7) for VLiM.

B. Performance metrics

The major objectives of a VNE problem are to maximize the InP revenues by accepting as many VNRs as possible, and to minimize the mapping cost. InP's revenue can be calculated as the sum of total virtual resources mapped on the corresponding SN over time, whereas the embedding cost of the i^{th} VNE $\Xi(G_i^v)$ is defined as the sum of total physical resources allocated to the i^{th} VN [6].

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

$$'\Upsilon(G_i^v) = w_\alpha * \sum_{l_i^v \in L_i^v} b(l_i^v) + w_\beta * \sum_{n_i^v \in N_i^v} c(n_i^v) \tag{8}$$

where $b(l_i^v)$ 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_α and w_β are the unit weights of the bandwidth and CPU resources respectively.

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

$$\Xi(G_i^v) = \sum_{n_i^v \in N_i^v} c(n_i^v) + \sum_{l_i^v \in L_i^v} \sum_{l^s \in L^s} f_{l^s}^{l_i^v} \tag{9}$$

where $f_{l^s}^{l_i^v}$ defines the bandwidth of substrate link l^s that is allocated to the virtual link l_i^v.

Acceptance ratio: is a ratio between the number of accepted VNRs over the number of arrived VNRs during an interval time τ is computed as following:

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

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

Node utilization indicates the distribution of workloads on the corresponding SN. Node utilization defines the amount of network resources occupied by virtual node requests during a certain time, divided by the total amount of node resources. Node utilization can be expressed as follows:

$$\mathcal{U}_N(N^s) = \sum_{n^s \in N^s} \left(\frac{\sum_{n_i^v \rightarrow n^s} c(n_i^v)}{c(n^s)} \right) * T_i \ , \tag{11}$$

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

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

$$\mathcal{U}_L(L^s) = \sum_{l^s \in L^s} \left(\frac{\sum_{l_i^v \rightarrow l^s} b(l_i^v)}{b(l^s)} \right) * T_i \ , \tag{12}$$

Fitness Function (FF) is utilized to evaluate the quality of VNE solutions. Fitness values can serve as rewards for guiding the searching process to the optimal solutions. In this paper, FF considers multiple factors including embedding cost, hop-count and propagation delay, determining the optimal mapping solution for a VNR. These factors reflect total network resources comprising both CPU and bandwidth as well as network latency. Our fitness function $\mathcal{F}(\mathcal{S})$ can be expressed as below:

$$\begin{aligned} \mathcal{F}(\mathcal{S}) = & \left(\frac{1}{\Upsilon(G_i^v)}\right) * w_c + \left(\frac{1}{\sum_{l_i^v \in L_i^v} h_{\mathcal{A}_{\mathcal{L}}(l_i^v)}}\right) * w_h \\ & + \left(\frac{1}{\sum_{l_i^v \in L_i^v} d_{\mathbb{P}}(\mathcal{A}_{\mathcal{L}}(l_i^v))}\right) * w_p \end{aligned} \tag{13}$$

where, $\mathcal{S}$, h and $d_{\mathbb{P}}$ are a feasible solution, hop-count and propagation delay of the link mapping solution of l_i^v respectively. w_c, w_h, and w_p are weight parameters equivalent to cost, hop-count and propagation delay.

III. Joint node-link Embedding Algorithm

GA is a AI algorithm that is able to solve constrained or unconstrained optimization problems. A GA algorithm typically comprises four primary operations: population initialization, selection, crossover and mutation. Our proposed parallel GA scheme is presented in Fig 1.

A. Sequential Path Searching Algorithm

Indeed, SN topology is almost static, which means that it is able to determine a number of physical paths for each pair of source and destination nodes in advance. In this paper, a path database for link mappings are extensively established before VNR arrivals. The shortest path mechanism (e.g., Dijkstra's algorithm) based on the hop-count factor is deployed to construct the path database which is recognized as the initial path pool generation in Fig. 1. Minimizing hop-count can improve the embedding cost and physical link utilization reflected by path length metric, leaving more resources to future VNRs to be accepted. The first

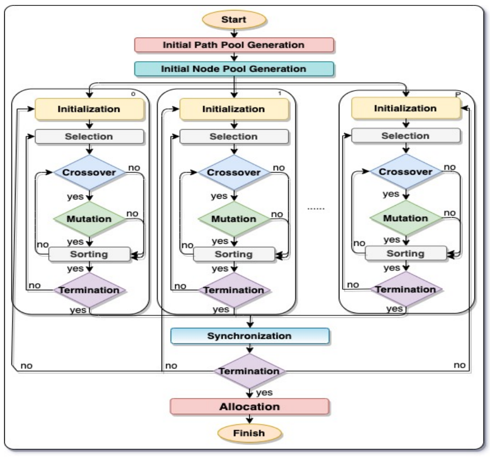


Fig. 1: *Parallel operation scheme*

path of each pair of source-destination nodes has minimum hop count. Additionally, by forwarding data to less nodes, overheads imposed by sending repetitive data and energy consumption can be improved. When a virtual link request arrives for mapping, our sequential path searching algorithm consecutively seeks for a feasible path in database based on the node mapping information. If the first path does not meet the resource requirements of the VNR (e.g., bandwidth), the algorithm will approach the consecutive one until at least one feasible path is achieved. As such, the feasible path possesses minimal hop-count feature. Otherwise, if there is not any feasible found, the mapping returns failed. Since the path database can be generated prior to the arrival of VNRs, time counted for this process can be neglected. In fact, this searching mechanism is simple, fast and efficient.

B. Conciliation Strategy

Population in GA contains multiple chromosomes that are generated in random manner. A given VNR typically consists of a set of virtual nodes connected by multiple virtual links. Consequently, there are two sets of mapping solutions for virtual nodes and virtual links. The node mappings apparently influence the later ones; consequently, if the node mappings are altered, the link solutions are correspondingly changed. As a result, our proposed GA algorithm defines the mapping solution of a virtual node as a gene. In this paper, virtual

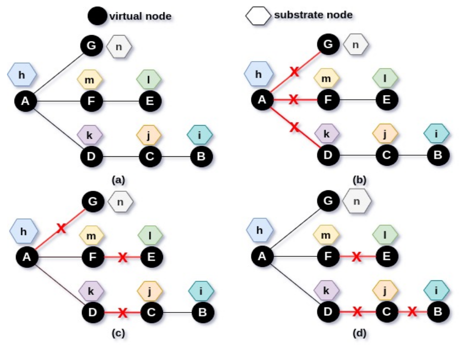


Fig. 2: *Examples of infeasible virtual link mappings*

nodes in a given VNR are embedded in order as similar to NFV-FGE, so their order in all chromosome is also the same. Towards improving the capability and reducing the execution time, a feasible node pool, in which all feasible substrate nodes satisfying the least CPU capacity requirement of the VNR are collected, is advised to be first created. Then, virtual nodes in a given VNR are assigned one by one to feasible substrate nodes that are selected from the initial node pool in random. In fact, the random VNoM can avoid the possible premature convergence problem of GA algorithm.

After the mappings of all virtual nodes in a VNR are identified, a sequential path searching method described in III-A is deployed to discover feasible link mappings for the virtual link requests thanks to the information of already-found node mappings. These substrate paths must meet all virtual resource demands (Eq. (5)-(7)) to become the potential link mapping solutions. When a feasible solution for the whole VNR has been successfully recognized, the chromosome can be established. Otherwise, if there is no feasible path (e.g., network congestion) found for a virtual link request, the pair of corresponding virtual nodes need to be remapped. The more virtual links are failed to map, the more complicated problem is. Instead of finding appropriate virtual nodes to remap, VNoM process can be started over again, but this approach is obviously inefficient and might overlook optimal solutions.

Algorithm 1 Heuristic Conciliation Algorithm

```
1: Input:
2:     Initial solutions of virtual node mapping
3:     A set of failed virtual links after link mapping phase
4: Output:
5:     Virtual nodes to be remapped
6: procedure HEURISTIC CONCILIATION STRATEGY
7:     Step 1: Construct a map $M_p$ of virtual nodes based on their presence in a set of failed virtual links
8:     Step 2: Create a map $M_n$ of virtual nodes based on their nearest neighbors ▷ These steps can be done in advance prior this algorithm
9:     Initialize an array for remapped nodes
10:    for each infeasible virtual link do
11:        if $s_i^v$ or $d_i^v$ NOT in the array then
12:            if $M_p[s_i^v] > M_p[d_i^v]$ then
13:                add $s_i^v$ into array
14:            else if $M_p[s_i^v] < M_p[d_i^v]$ then
15:                add $d_i^v$ into array
16:            else
17:                if $M_n[s_i^v] > M_n[d_i^v]$ then
18:                    add $d_i^v$ into array
19:                else
20:                    add $s_i^v$ into array
21:                end if
22:            end if
23:        end if
24:    end for
25:    return node array
26: end procedure
```

We aim at minimizing the number of virtual nodes associated with the failed-mapping virtual links that should be re-embedded. Let's take an example with a specific VNR with 7 virtual nodes and 6 virtual links as illustrated in Fig. 2. Fig 2a in the top-left is a complete situation as all virtual nodes and links are explored their mappings successfully. However, next figures demonstrate some “real” problems of link mapping failures. For example, there are three failed virtual link requests $\{$A - G, A - F and A - D$\}$ according to the node mappings $\{A \to h, G \to n, F \to m$ and $D \to k\}$ as shown in Fig. 2b. Unfortunately, no feasible paths are found between $\{h \to n, h \to m$, and $h \to k\}$. The problem is that if we revisit all already-mapped virtual nodes (definitely ineffective) or virtual nodes {D, F and G}, remapping at least five virtual links is required. Otherwise, if the virtual node A, is only re-embedded, the virtual links that need to be remapped are merely three. In similarity, virtual nodes C/D, E and G can be revisited in Fig. 2 and virtual nodes C and E should be reconsidered in the last figure by cause of least remapping. Motivated by this idea, we propose a heuristic conciliation algorithm to deal with this issue as described in Algorithm 1.

C. Working machines

Population Initialization: As shown in Fig. 1, every working machine running GA algorithm begins with a population initialization operation. $\mathcal{M}$ is the set of chromosomes where each chromosome consists of $\mathcal{N} = |N_i^v|$ genes considered as the potential mappings of all virtual nodes of the VNR. The initial population $\mathcal{P}$ ($\mathcal{M}$x$\mathcal{N}$) at a working machine is established as elucidated in Section III-B. As soon as a chromosome is constructed, the sequential path searching mechanism (Section III-A) based on the already-achieved node mappings is implemented for VLiM stage.

$$\mathcal{P} = \begin{bmatrix} \mathcal{C}_1 \\ \mathcal{C}_2 \\ \vdots \\ \mathcal{C}_f \\ \vdots \\ \mathcal{C}_{\mathcal{M}} \end{bmatrix} = \begin{bmatrix} g_1^1 & \cdots & g_1^j & \cdots & g_1^{\mathcal{N}} \\ g_2^1 & \cdots & g_2^j & \cdots & g_2^{\mathcal{N}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ g_f^1 & \cdots & g_f^j & \cdots & g_f^{\mathcal{N}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ g_{\mathcal{M}}^1 & \cdots & g_{\mathcal{M}}^j & \cdots & g_{\mathcal{M}}^{\mathcal{N}} \end{bmatrix} \tag{14}$$

Thereupon, a conciliation algorithm addresses possible failures of virtual link mappings, minimizing remapping virtual node/link mappings. When feasible solutions of all virtual nodes and links are achieved, the chromosome of a given VNR is eventually formed. It is noted that any change on the node mappings leads to the associated changes of link mappings.

Evolved generations: Random chromosomes in population are selected to become parents for producing their children or next generations as the results of crossover and mutation operators. The fast and efficient path searching method is deployed to approach link mappings for new generated offspring. The population will update these new generations to strengthen the population diversity.

Crossover combines parental chromosomes to generate new offspring.

$$\mathcal{P} = \begin{bmatrix} \mathcal{C}_1 \\ \vdots \\ \mathcal{C}_s \\ \vdots \\ \mathcal{C}_r \\ \vdots \\ \mathcal{C}_{\mathcal{M}} \\ \mathcal{C}_{\mathcal{M}+1} \\ \mathcal{C}_{\mathcal{M}+2} \end{bmatrix} = \begin{bmatrix} g_1^1 & \cdots & g_1^{j^c} & g_1^{j^c+1} & \cdots & g_1^{\mathcal{N}} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ g_s^1 & \cdots & g_s^{j^c} & g_s^{j^c+1} & \cdots & g_s^{\mathcal{N}} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ g_r^1 & \cdots & g_r^{j^c} & g_r^{j^c+1} & \cdots & g_r^{\mathcal{N}} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ g_{\mathcal{M}}^1 & \cdots & g_{\mathcal{M}}^{j^c} & g_{\mathcal{M}}^{j^c+1} & \cdots & g_{\mathcal{M}}^{\mathcal{N}} \\ g_s^1 & \cdots & g_s^{j^c} & g_r^{j^c+1} & \cdots & g_r^{\mathcal{N}} \\ g_r^1 & \cdots & g_r^{j^c} & g_s^{j^c+1} & \cdots & g_s^{\mathcal{N}} \end{bmatrix} \tag{15}$$

Denote $\mathcal{C}_s$ and $\mathcal{C}_r$ as parental chromosomes indexed s and r in the initial population. Their new descendants are represented as $\mathcal{C}_{(\mathcal{M}+1)}$ and $\mathcal{C}_{(\mathcal{M}+2)}$ respectively, where j^c is the random crossover point between any genes within $|\mathcal{N}|$. In crossover, offspring is produced by swapping genes between parents from the starting point $j^c + 1$ to $|\mathcal{N}|$ as depicted in (15).

Mutation usually applies a modification on an individual parent for delivering new offspring. This operation is aimed at sampling the broad solution space as well as enhancing the efficiency of solution exploration. Mutation also avoids potential solutions falling into the local optima. Denote j^m and $g_{r'}^{j^m}$ as the random mutation point and new gene substituting the existing one in $\mathcal{C}_{(\mathcal{M}+1)}$, respectively. New solution $\mathcal{C'}_{(\mathcal{M}+1)}$ after substitution is defined as $\mathcal{C'}_{(\mathcal{M}+1)} = [g_s^1 \cdots g_{r'}^{j^m} \cdots g_s^{\mathcal{N}}]$.

D. Sorting and Terminations

The mapping solutions are sorted to select the best one relied on their higher fitness values. It is then delivered to synchronization for the global ranking. Waiting for every worker machine accomplishing its assigned task is troublesome, without guaranteeing an expected outcome. Hence, the master node aborts GA algorithms running in working machines if no better mapping is obtained within t times, where t denotes a termination parameter.

E. Synchronization and VNR allocation

In this process, the best mappings of the given VNR is recognized by globally ranking the solutions received from working machines, based on highest fitness values. Finally, network updates allocated resources.

IV. Performance Evaluation

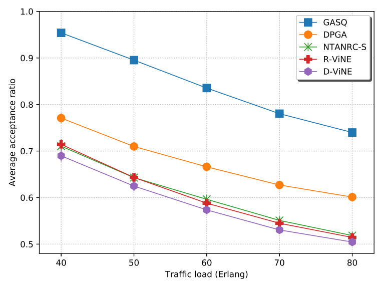
Our proposed GA-based algorithm, namely ***GASQ***, is compared with several state-of-the-art VNE competitors including DPGA [7], NTANRC-S [9], R-ViNE and D-ViNE [6] on various performance metrics.

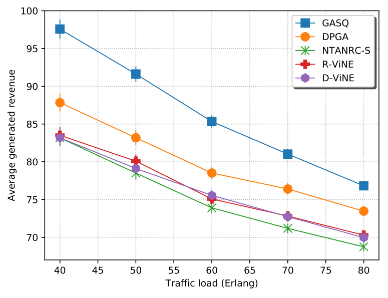
A. Simulation setup

We have developed a discrete-event simulator to evaluate the GASQ algorithm with similar parameters set out in [6]. The popular GT-ITM topology generator [10] is deployed to generate SNs and VNs, where SNs are configured with average 50 nodes and 140 edges adopting Waxman model ($\alpha = 0.5$ and $\beta = 0.2$). CPU and bandwidth capacity of SNs are uniformly generated between 50 and 100 units, whereas VNRs arrive dynamically 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 average $\mu = 1000$ time units, so the workload of VNRs can be quantified by $\frac{\lambda}{\mu}$ Erlangs. Towards VNs, the number of virtual nodes in each VNR is uniformly distributed between 2 and 10, where CPU and bandwidth requirements of VNRs are uniformly generated between 0 to 20 and 0 to 50 respectively. Each simulation runs $50,000$ time units, 50 times longer than the average lifetime of a VN, exceptionally generating a large number of independent samples. All performance figures are plotted with average values with 95% confidence interval.

B. Evaluation Results

As illustrated in Fig 3a, GASQ algorithm outperformed all compared algorithms in terms of acceptance ratio, resulting to highest average revenue which is the most desired objective for any embedding approach. Our proposed algorithm also proved its efficiency by achieving the lowest cost (Fig. 3c). Particularly, GASQ performed better acceptance ratios than DPGA for more than 19.02% and 13.62% at 40 and 80 Erlang respectively. GASQ also outperformed NTANRC-S,


(a)


(b)

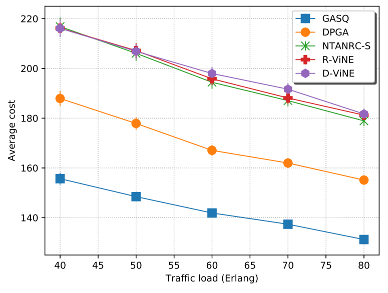
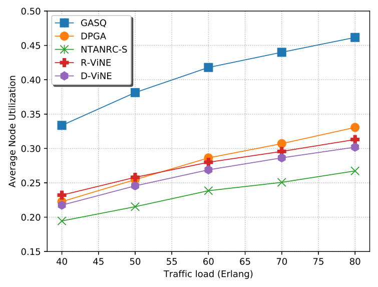
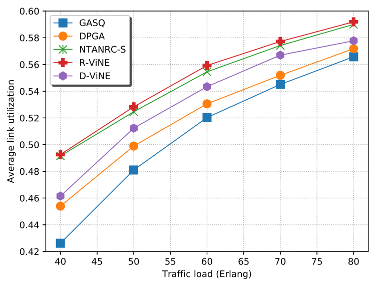

(c)

Fig. 3: **(a)** Average Acceptance Ratio **(b)** Average generated revenue **(c)** Average mapping cost


(a)


(b)

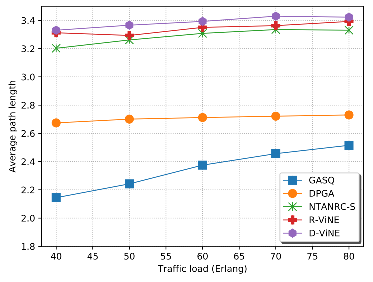

(c)

Fig. 4: **(a)** Average node utilization **(b)** Average link utilization **(c)** Average path length

R-ViNE and D-ViNE for more than 20% at the highest traffic load. Our joint node and link approach improved average revenue and cost up to 11.77% and 27.78% compared to the aforementioned algorithms at the same traffic load respectively. Furthermore, the node utilization of the proposed algorithm was much better than all competitors ranging from 10.15% up to 18.93% at various traffic loads as shown in Fig. 4a. Fig. 4b and 4c indicated that GASQ and DPGA had average path lengths shorter than the other algorithms, contributing to lower embedding cost and improved average link utilization. Thanks to less bandwidth consumption, a plethora of remaining bandwidth allowed to accept more arriving VNRs. The remarkable results were of the efficient sequential path searching based on hop-count factor in a novel joint node-link approach and the multiple constrained FF driving the GA algorithm to approach the optimal mapping solutions. Our proposed algorithm successfully solved the lack of coordination between node and link mappings, which leads to an ultimate embedding outcome that could be a tough challenge for any future VNE algorithm. Additionally, we compared execution time of our VNE approach on both sequential and parallel schemes in order to quantify time reduction of parallel operation. Due to the distributed parallel paradigm, GASQ needed $1.19s$ to allocate a VNR whereas the sequential counter part finished the same task in $9.73s$. Regarding time complexity, interested readers may refer to the paper [7] for further theoretical analysis.

V. Conclusion

This paper proposes a joint node and link GA-based embedding algorithm for dealing with the VNE problem in one stage. A novel heuristic conciliation algorithm is utilized to handle multiple infeasible link mappings due to the inappropriate node embedding in GA operations. We also advise a distributed parallel operation scheme to reduce execution time of GA algorithm. A comparison between sequential and parallel operations of our proposed solution is consequently provisioned. The extensive evaluation indicates that the joint node and link approach based on GA algorithm in a single embedding stage outperforms state-of-the-art heuristic VNE algorithms in all indispensable performance metrics we adopted.

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] 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.

[4] 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

[5] 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

[6] 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.

[7] 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.

[8] 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.

[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] 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.