Joint Node-Link Algorithm for Embedding Virtual Networks with Conciliation Strategy

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*—Network virtualization (NV) has widely envisioned as a crucial factor for the success of the future networks by enabling a flexible, cost-effective and on-demand deployments of multiple network service requests on a shared physical infrastructure. The major challenge of NV is to efficiently and effectively embed heterogeneous virtual network requests (VNRs), consisting of 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. The lack of a coordination between node and link mapping stages results in low acceptance ratio as well as network revenues. In this paper, we propose a new approach based on Genetic Algorithm (GA) that jointly coordinates node and link mappings where the link mapping is relied on a path ranking method. A novel heuristic conciliation mechanism is introduced to handle a possible set of infeasible link mappings during generating virtual embedding solutions in GA's operations. Extensive evaluation results show that our proposed GA-based algorithm outperforms state-of-the-art virtual embedding algorithms in all performance metrics we adopted.**

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

I. Introduction

NV is emerged as a de facto paradigm for the foreseen success of future networks such as virtualized fifth generation [1], smart Internet of Things (IoT) networks [2]. The adoption of NV enables to share physical resources amongst multiple VNRs, providing an isolated coexistence of a number of VNs on a shared underlying substrate network (SN). In virtualization environment, a network service provider (SP) commonly converts a service or application into a VN which is then transferred to a corresponding infrastructure provider (InP) as a form of a VNR. The InP attempts to map such VN onto its substrate infrastructure meeting several rigorous resource constraints by adopting an optimization process. Mapping VNRs onto the shared underlying physical infrastructure with varied topology and stringent resource demands is known as a virtual network embedding (VNE) problem. VNE is widely known as a generalization of second stage of network function virtualization resource allocation (NFV-RA), namely virtual network function forwarding graph embedding (NFV-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 [3]. NFV-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 [3]. Additionally, VNE is assumed to be more complicated than NFV-FGE in some specific cases and topology [4], [5]. VNE is proven to be $\mathcal{NP}$-Hard either for VNoM or VLiM [3], [6]. Optimization models (e.g. Integer Linear Programming (ILP)) are formulated to achieve optimal VNE solutions, but there are several challenges since these models have to confront with scalability, complex deployment, and high time complexity. Exact solutions are not indeed tailored for online VNE problems. Hence, most of research papers have concentrated on efficient designs of heuristics or meta-heuristic algorithms to tackle the aforementioned obstacles of the optimization models.

Similar to NFV-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 [7]. The decoupling can facilitate the complexity of algorithmic deployment, 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. For instance, although VNoM stage can figure out promising substrate nodes for mapping virtual nodes in a VNR, there may not be any paths in the substrate network that have enough resources to support the request for virtual links connecting those virtual nodes. Then, the corresponding VNR is obviously rejected. In practise, the most common failures of mapping VNRs invariably emanate from the ineffective link mapping algorithm [8]. In this paper, we propose a GA-based algorithm to solve VNE problem by jointly mapping virtual nodes and corresponding virtual links. GA attempts to find potential solutions for all virtual nodes in a VNR. Then, corresponding virtual links will be sequentially embedded based on a novel path ranking method which allows to select the best virtual link mappings subject to multiple objectives. If no feasible paths are found for some link requests due to network congestion, the current mapping solutions of virtual nodes should be reconsidered with minimal node and link mappings changed. As a result, we introduce an efficient conciliation algorithm to deal with this problem. Furthermore, we present a distributed parallel GA-based scheme that exploits 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 achieve better resource utilization in theory, but this solution can generate abundant overhead for maintaining consistent network state and possibly results in out-of-order

delivery problem with extra latency that might be unacceptable for sensitive-delay applications [9]. Thus, we only consider unsplittable-enabled mapping solutions in this paper.

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 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 $loc(n_i^v)$. In this paper, our objectives are to maximize average acceptance ratio, average revenue over cost ratio and to improve node and link mapping and propagation delay, satisfying node and link constraints imposed by VNRs.

Node constraints:

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

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

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

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

where $\mathcal{A}_\mathcal{N}(n_i^v)$ is mapping solution of the virtual node. $\mathcal{D}(i^s, j^d)$ and $R_N(n^s)$ denote the distance between i^s and j^d, and the remaining CPU capacity of a substrate node respectively.

Link constraints:

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

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

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

where $\mathcal{E}^s(\mathcal{A}_\mathcal{L}(l_i^v))$ denotes a set of all available paths from the source $\mathcal{A}_\mathcal{N}(s_i^v)$ to destination node $\mathcal{A}_\mathcal{N}(d_i^v)$. $R_L(e^s)$ 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. A mapping solution is called "feasible" if it meets resource constraints (1)-(4) for node mapping or (5)-(7) for link mapping. A path for a mapped virtual link in the feasible solution is therefore called a feasible path.

B. Performance metrics

Main objectives of a VNE problem are maximizing the revenues of InP while minimizing the corresponding embedding cost. The ratio of average revenue over cost can be used to evaluate the efficiency of any VNE algorithm. In this paper, InP's revenue is calculated as the sum of total virtual resources mapped on the SN over time whereas the embedding cost of the i^{th} VNE $C(G_i^v)$ is the sum of total network resources allocated to the i^{th} VN.

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

$$\Upsilon(G_i^v) = \frac{\mathcal{R}(G_i^v)}{C(G_i^v)} = \frac{w_b * \sum\limits_{l_i^v \in L_i^v} b(l_i^v) + w_n * \sum\limits_{n_i^v \in N_i^v} c(n_i^v)}{\sum\limits_{n_i^v \in N_i^v} c(n_i^v) + \sum\limits_{l_i^v \in L_i^v} \sum\limits_{l^s \in L^s} f_{l^s}^{l_i^v}} \tag{8}$$

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 are the unit weights of the bandwidth and CPU resources respectively. Besides, $f_{l^s}^{l_i^v}$ 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:

$$\mathcal{A}_c^\tau = \left| \frac{\xi^a(\tau)}{\xi(\tau)} \right| \tag{9}$$

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

Node utilization reflects the distribution of network loads on the corresponding SN. Node utilization is derived from 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:

$$\mathcal{U}_N(N^s) = \sum_{n^s \in N^s} \left(\frac{\sum\limits_{n_i^v \to n^s} c(n_i^v)}{c(n^s)} \right) * T_i \quad , \tag{10}$$

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

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

$$\mathcal{U}_L(L^s) = \sum_{l^s \in L^s} \left(\frac{\sum\limits_{l_i^v \to l^s} b(l_i^v)}{b(l^s)} \right) * T_i \quad , \tag{11}$$

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 solution. In this paper, FF takes the mapping cost, hop-count and propagation delay into consideration and decides which the best solution for a VNR is. 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. Our multi-objective fitness function $\mathcal{F}(\mathcal{S})$ can be expressed as below:

$$\begin{aligned} \mathcal{F}(\mathcal{S}) = \left(\frac{1}{C(G_i^v)} \right) * w_c + \left(\frac{1}{\sum_{l_i^v \in L_i^v} h_{\mathcal{A}_\mathcal{L}(l_i^v)}} \right) * w_h \\ + \left(\frac{1}{\sum_{l_i^v \in L_i^v} d_\mathbb{P}(\mathcal{A}_\mathcal{L}(l_i^v))} \right) * w_p \end{aligned} \tag{12}$$

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 factors.

III. Joint Node-Link Mapping

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. Our proposed parallel GA scheme is presented in Fig 1. Each working machine independently

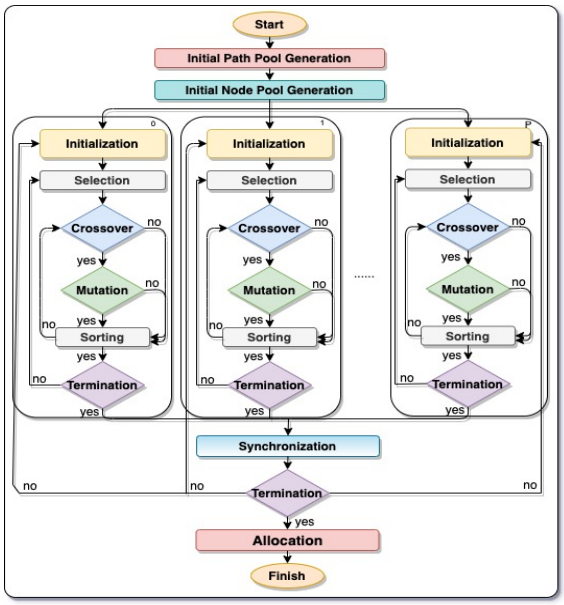


Fig. 1: *Parallel operation scheme*

runs GA algorithm with a pre-defined number of iterations to explore feasible solutions for a VNR. The best-matching outcome is selected among these parallel machines. When node mappings are changed, the link mappings altered accordingly.

A. Path Ranking Algorithm

We argue that substrate paths for each pair of source and destination nodes in a SN can be determined in advance due to the fact that topology of a SN is basically static. Consequently, initial path database for VLiM can be constructed completely prior to the arrival of online VNRs. The shortest path method (e.g., Dijkstra's algorithm) focusing on hop-count feature is used to build this database, called initial path pool generation in Fig. 1. When there is a virtual link request, the path ranking algorithm sequentially finds a predefined number of feasible paths in database for each source-destination pair (e.g., r = 10) based on the node mapping information of such request given. This number is large enough to guarantee a good mapping for any link request and it can be adjusted to compromise a trade-off between expected performance and execution time. These paths can then be ranked by the multi-objective fitness function (Eq. 12) to select the best mapping solution for the virtual link request.

B. Conciliation Construction

In GA's operations, population 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 two separate sets of solutions for virtual nodes and virtual links. The later solutions are dependent on the former ones, so if the node mappings are changed, the link solutions are altered correspondingly. Hence, our proposed GA algorithm defines a specific mapping of a virtual node as a gene. A chromosome $\mathcal{C}_f$ is a specific mapping of all the virtual nodes of a VNR. Each gene g_f^j is associated with a mapping solution of a virtual node, where f and j indicate the chromosome f and virtual node j in a VNR respectively. We consider all virtual node mappings are similar. 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 virtual node requirements 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, meeting resource requirements of the virtual node (Eq. (1)-(4)). We assume that mapping virtual nodes is conducted in random to prevent the possible premature convergence problem.

When mappings of all virtual nodes are determined, a path ranking method as described in III-A is applied to explore feasible mapping solutions of associated virtual link requests, based on the already-found node mapping information. These physical paths must meet virtual resource requirements to become potential link mapping solutions (Eq. (5)-(7)). Since a feasible solution for the whole VNR has been successfully defined, a chromosome will be established. If a virtual link request cannot find any feasible path (e.g., network congestion), such pair of virtual nodes need to be remapped. We would like to minimize the number of virtual nodes

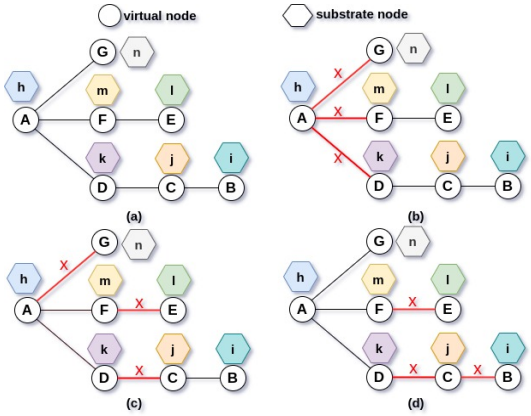


Fig. 2: *Examples of infeasible virtual link mappings*

associated with the corresponding failed-mapping virtual links that should be remapped. Let us take an example with a specific VNR towards 7 virtual nodes and 6 virtual links as shown in Fig. 2. Fig 2a is an ideal situation when all virtual nodes and links have been successfully determined their embedding solutions. Next figures illustrate some possible failures of virtual link mappings. In Fig. 2b, we can see that there are three infeasible link mappings of three virtual link requests including {A - G, A - F and A - D}. With node embedding solutions $\{A \rightarrow h, G \rightarrow n, F \rightarrow m$ and $D \rightarrow k\}$, there are no feasible paths found between $\{h \rightarrow n, h \rightarrow m$, and $h \rightarrow k\}$. 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 virtual node A, is revisited, the remapped virtual links could be three only.

Similarly, virtual nodes C/D, E and G should be reconsidered in Fig. 2 while virtual nodes C and E can be remapped in the last figure due to least remapping. Inspired by this idea, we present a heuristic conciliation algorithm to handle this problem as detailed in Algorithm 1. In this paper, the conciliation is now considered in GA algorithm to solve the VNE problem.

Algorithm 1 Heuristic Conciliation Algorithm

Input:
  Initial solutions of virtual node mapping
  A set of failed virtual links after link mapping phase
Output:
  Virtual nodes should be remapped
procedure HEURISTIC CONCILIATION STRATEGY
  Step 1: *Construct a map M_p of virtual nodes based on their presence in a set of failed virtual links*
  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
  Initialize an array for remapped nodes
  for each infeasible virtual link **do**
    if s_i^v or d_i^v NOT in the array **then**
      if $M_p[s_i^v] > M_p[d_i^v]$ **then**
        add s_i^v into array
      else if $M_p[s_i^v] < M_p[d_i^v]$ **then**
        add d_i^v into array
      else
        if $M_n[s_i^v] > M_n[d_i^v]$ **then**
          add d_i^v into array
        else
          add s_i^v into array
        end if
      end if
    end if
  end for
  return *node array*
end procedure

C. Working node

Population Initialization: Each working machine starts with a population initialization step. Denote $\mathcal{M}$ as a set of chromosomes. Each chromosome includes $\mathcal{N} = |N_i^v|$ genes that represent potential mapping solutions of all virtual node requests in a VNR. An initial population $\mathcal{P}$ ($\mathcal{M}$x$\mathcal{N}$ size) at a working machine is generated as described in Section III-B.

$$\mathcal{P} = \begin{bmatrix} \mathcal{C}_1 \\ \mathcal{C}_2 \\ \vdots \\ \mathcal{C}_f \\ \vdots \\ \mathcal{C}_\mathcal{M} \end{bmatrix} = \begin{bmatrix} g_1^1 & \cdots & g_1^j & \cdots & g_1^\mathcal{N} \\ g_2^1 & \cdots & g_2^j & \cdots & g_2^\mathcal{N} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ g_f^1 & \cdots & g_f^j & \cdots & g_f^\mathcal{N} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ g_\mathcal{M}^1 & \cdots & g_\mathcal{M}^j & \cdots & g_\mathcal{M}^\mathcal{N} \end{bmatrix} \tag{13}$$

As such, each chromosome in population is initially formed and then the path ranking mechanism is deployed for VLiM based on node mapping solutions that have been already achieved. A conciliation mechanism is used to handle possible failures of virtual link mappings with a goal of minimal virtual node/link mappings revisited. Since feasible solutions of all virtual nodes and links are successfully obtained, the chromosome for a VNR is properly established. It appears that if any node mapping changes, the associated link mapping is affected accordingly.

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 these operations causing new generated nodes, the path ranking method in Section III-B is implemented to explore link mapping solutions. If no feasible paths are found, new generations are discarded; otherwise, they will be updated into the population to enhance the population diversity, increasing the possibility of approaching near-optimal solutions. Thus, we set both crossover and mutation probabilities equal to 0.9 in this paper.

Crossover: This procedure combines parental chromosomes to generate new offspring for next generations. $\mathcal{C}_s$ and $\mathcal{C}_r$ denote two parental chromosomes with their indexes s and r in initial population, whereas new descendant chromosomes are described as $\mathcal{C}_{(\mathcal{M}+1)}$ and $\mathcal{C}_{(\mathcal{M}+2)}$ respectively. j^c indicates a random crossover point between any genes within $\mathcal{N}$ length. In crossover operation, offspring is established by swapping genes between parents starting from the crossover point j^c+1 to the end of the chromosomes as illustrated in (14). At this time, node mappings have been changed, link mappings are revisited accordingly thanks to the path ranking method. If new generations are feasible, they will be updated into population.

$$\mathcal{P} = \begin{bmatrix} \mathcal{C}_1 \\ \vdots \\ \mathcal{C}_s \\ \vdots \\ \mathcal{C}_r \\ \vdots \\ \mathcal{C}_\mathcal{M} \\ \mathcal{C}_{\mathcal{M}+1} \\ \mathcal{C}_{\mathcal{M}+2} \end{bmatrix} = \begin{bmatrix} g_1^1 & \cdots & g_1^{j^c} & g_1^{j^c+1} & \cdots & g_1^\mathcal{N} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ g_s^1 & \cdots & g_s^{j^c} & g_s^{j^c+1} & \cdots & g_s^\mathcal{N} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ g_r^1 & \cdots & g_r^{j^c} & g_r^{j^c+1} & \cdots & g_r^\mathcal{N} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \\ g_\mathcal{M}^1 & \cdots & g_\mathcal{M}^{j^c} & g_\mathcal{M}^{j^c+1} & \cdots & g_\mathcal{M}^\mathcal{N} \\ g_s^1 & \cdots & g_s^{j^c} & g_r^{j^c+1} & \cdots & g_r^\mathcal{N} \\ g_r^1 & \cdots & g_r^{j^c} & g_s^{j^c+1} & \cdots & g_s^\mathcal{N} \end{bmatrix} \tag{14}$$

Mutation: This operation typically adopts a modification on an individual parent to produce new offspring. Mutation samples the broad solution space and improves the searching efficiency, preventing solutions from falling into the local optima. A random gene of the selected parent is replaced by a new gene to produce a new child. The new gene must explicitly meet the resource constraints to be chosen. Similar to crossover, path ranking mechanism is implemented to find new link mappings due to node mappings changed. If feasible, new generation is updated into population. Let denote j^m and $g_{r'}^{j^m}$ as a random mutation point and new gene that replaced the existing one in $\mathcal{C}_{(\mathcal{M}+1)}$, respectively. In mutation, new embedding solution $\mathcal{C'}_{(\mathcal{M}+1)}$ after replacement can be presented as $\mathcal{C'}_{(\mathcal{M}+1)} = [g_s^1 \cdots g_{r'}^{j^m} \cdots g_s^\mathcal{N}]$.

D. Sorting and Terminations

Sorting process selects 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. To reduce execution time, the master node terminates GA algorithms in running worker nodes if no better mapping solution is achieved within t times, where t denotes a termination parameter.

E. Synchronization and VNR 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 VNR is then allocated onto SN following the information of the virtual node and link mapping solutions obtained. The last step is updating residual network resources.

IV. Performance Evaluation

We compare our proposed algorithm with several state-of-the-art VNE competitors: DPGA [7], NTANRC-S [10], D-ViNE, R-ViNE, and G-SP [6] on various performance metrics including average acceptance ratio, average revenue to cost ratio, average node and link utilization and average delay. Our simulation is conducted on a Ubuntu 20.04.2 LTS 64-bit platform with 15.5 GiB memory and Intel Core i5-6200U CPU@2.30GHz×4.

A. Simulation setup

We develop a discrete-event simulator to evaluate our proposed VNE solution with parameters similar to those in [6]. The popular GT-ITM topology generator [11] is used to generate SNs and VNs. SNs are configured with average 50 nodes randomly placed on a 25×25 Cartesian plane. They include average 140 edges adopting Waxman model with $\alpha = 0.5$ and $\beta = 0.2$, where α and β indicate the maximal edge probability and edge length respectively. 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. The miscellaneous loads 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. By generating arbitrary SNs and VNs for evaluation, our proposed algorithm can be adaptive to any topologies (e.g., datacenters) with diverse traffic patterns. In this paper, we set $w_b = w_n = 1$ as similar to [12] and $w_p = 0.4, w_c = 0.5$ and $w_h = 0.1$. Due to ingenious strategy, we define population size and the number of iterations merely equal to 10 and 15, respectively for GA algorithm in each parallel machine shown in Fig. 1. Each simulation runs $50,000$ 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

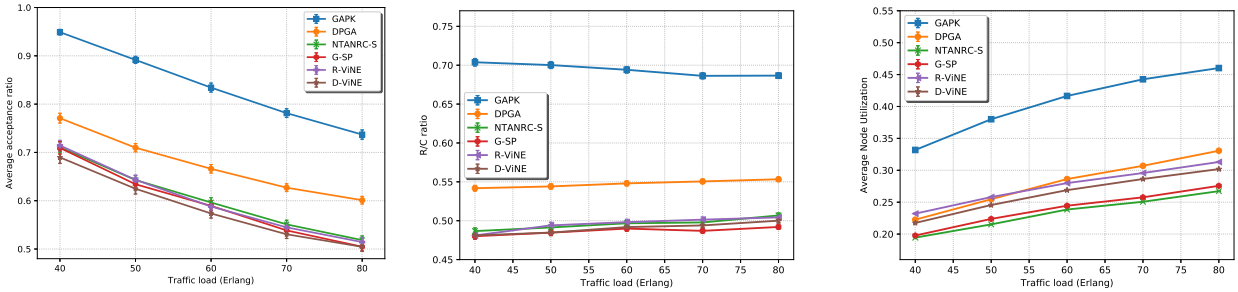
Simulation results are intensively illustrated in Figure 3 and 4. Our proposed GA-based algorithm, namely ***GAPK***, achieved highest acceptance ratio by accepting more VNRs with considerably lower costs than all competitors, leading to highest average revenue to cost ratio as illustrated in Fig 3b. In fact, higher acceptance and revenue to cost ratios is desirable for any VNE algorithm, so our proposed solution has proved its efficiency. Specifically, GAPK improved the acceptance ratios of DPGA for more than 17.81% and 13.59% at 40 and 80 Erlang respectively. GAPK significantly performed better than NTANRC-S and R-ViNE (the best algorithm in [6]) for more than 22% at the highest traffic load as shown in Fig. 3a. Our proposed joint node-link VNE solution gained $24.8\%, 34.4\%$ and 36.1% better average revenue to cost ratios than those of aforementioned algorithms at the same traffic load as depicted in Fig 3b. By accordingly accepting more VNRs, node utilization of GAPK is remarkably higher 10% up to 20% than its all rivals at all traffic load (Fig. 3c).

Moreover, it is observed that GAPK and DPGA extremely produced short average path length in a comparison with other algorithms as illustrated in Fig. 4b, which indeed contributes to low embedding cost and enhanced average link utilization (Fig. 4a). The reason behind these appealing outcomes is that we take hop-count factor into account in a multi-constrained FF for selecting the optimal VNE solution. Due to less bandwidth consumed to map virtual link requests, abundant residual bandwidth enables to accept incoming VNRs confirmed by Fig. 3a. In addition, we evaluate all compared algorithms in average delay metric. As depicted in Fig. 4c, GAPK empowered by an intelligent GA algorithm effectively explored physical paths with very low delay, reducing network resource fragmentation by preferring neighboring substrate nodes for mapping VNs.

Furthermore, we conducted the joint node-link VNE mapping on sequential and parallel schemes and then compared execution time between them to quantify time reduction with parallel operation. Distributed parallel paradigm with maximum 16 parallel machines as shown in Fig.1 took $1.2s$ to finish processing a VNR in average while the sequential counter part needed more than $10.1s$ to accomplish the same task. With regard to time complexity and convergence, interested readers may refer to the paper [7] for further theoretical analysis.

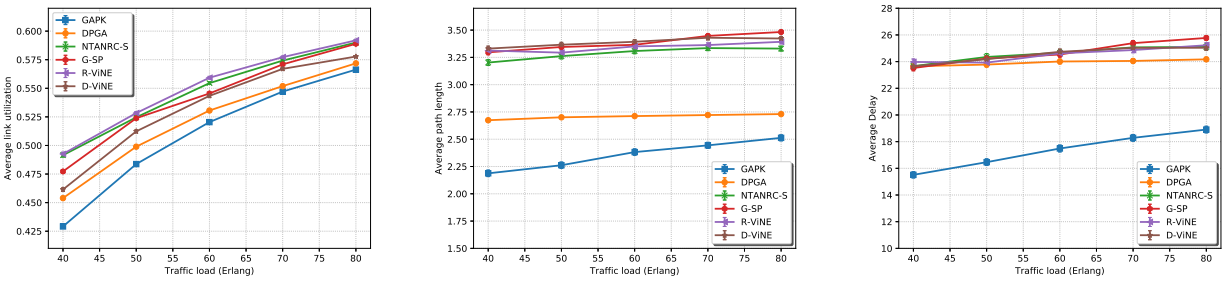
V. Related Work

VNE problem is widely known as $\mathcal{NP}$-hard in nature, that is intractable to be solved with Integer Programming (LP). Hence, a large number of research papers have focused on efficient heuristic algorithms due to high time complexity of extract methods. [6] proposed 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 [13] was an extension of [6] with a novel node splitting and node collocation approaches. Authors in [14] presented a topology-aware node mapping based on the Markov Random Walk model to quantify network resources. The research in [15] studied various topological attributes for enhancing a coordinate node-link mapping and then introduced different node-ranking algorithms. Zhang et al. [16] 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. Nguyen et al. [7] presented a novel GA-based algorithm for VLiM stage, which validated the important role of VLiM in VNE problem. Recently, reinforcement learning algorithms have been studied to deal with VNE problems in [17]. However, the aforementioned VNE papers



(a) (b) (c)

Fig. 3: **(a)** VNR Acceptance Ratio **(b)** Average revenue to cost ratio **(c)** Average node utilization



(a) (b) (c)

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

only focus on mapping VNRs in separate stages, including our previous work in [7] where virtual link mapping was the core. This paper is aimed at combining node and link mapping. A path ranking method and conciliation mechanism for coordinated virtual link mapping are also proposed.

VI. Conclusion

There are very few papers dealing with online VNE problem using heuristic or metaheuristic algorithms in a joint manner. In this paper, we propose joint node-link embedding approach based on GA algorithm for simultaneously solving virtual node and link mappings in one stage. A heuristic conciliation mechanism is deployed in GA operation to handle several infeasible link mappings due to improper virtual node embedding. Moreover, we present a distributed parallel operation scheme to reduce time complexity of GA algorithm. A comparison between sequential and parallel operation of the proposed VNE solution is accordingly provided. Our extensive evaluation shows that joint node-link combination in a single VNE mapping stage based on GA algorithm outperforms state-of-the-art heuristic VNE algorithms in all performance metrics we adopted.

References

[1] A. Hakiri and P. Berthou, "Leveraging sdn for the 5g networks: Trends, prospects and challenges," *ArXiv*, vol. abs/1506.02876, 2015.

[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.

[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.

[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] 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.

[10] 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.

[11] 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.

[12] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862 – 876, 2010.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.